pywws Documentation

Release 18.04.0.dev1387

Jim Easterbrook

03 apr 2018

Indice

Python software for USB Wireless WeatherStations.

pywws is a collection of Python modules to read, store and process data from popular USB wireless weather stations such as Elecsa AstroTouch 6975, Watson W-8681, WH-1080PC, WH1080, WH1081, WH3080 etc. I assume any model that is supplied with the EasyWeather Windows software is compatible, but cannot guarantee this.

The software has been developed to run in a low power, low memory environment such as a router or Raspberry Pi. It can be used to create graphs and web pages showing recent weather readings, typically updated every hour. It can also send «live» data to services such as Weather Underground and post messages to Twitter.

The development version of pywws is hosted on GitHub.

- https://github.com/jim-easterbrook/pywws

«Snapshot» releases of pywws are available from the Python Package Index (PyPI).

- https://pypi.python.org/pypi/pywws

Documentation is hosted on Read the Docs.

- http://pywws.readthedocs.org/

Documentation is available in the following languages (non-English versions may not be complete or up to date):

- English
- Français – translated by Jacques Desroches
- Italiano – translated by Edoardo

Ho scritto il programma per le mie necessità, ma ho fatto in modo che sia adattabile alle necessità altrui. Voi potete modificare alcuni o tutti i moduli, o scriverne dei nuovi, per ottenere esattamente quello che desideri.La ragione per cui è stato scelto Python è che rende le modifiche facili. Non abbiate paura, mettetevi alla prova..

CAPITOLO 1

Requisiti

Il software necessario per eseguire pywws dipende da cosa si intende fare con esso. È necessario Python 2.5 o successiva – Python 3 è parzialmente supportato, alcune funzionalità dipendono dalle librerie che non sono ancora state portate in Python 3.

Per ulteriori dettagli, vedi *Dipendenze*.

CAPITOLO 2

Installing and upgrading pywws

pywws can be installed directly from the Python Package Index (PyPI) using the pip command. See *Come iniziare con pywws* for full instructions.

Some new versions of pywws change what's stored in the hourly, daily or monthly summary data files. These new versions are incompatible with processed data from earlier versions. The `pywws.Reprocess` script regenerates all the summary data. It should be run after any major upgrade.

CAPITOLO 3

Documentazione

La documentazione è inclusa con pywws ed è anche disponibile online. Un buon punto di partenza è *Come iniziare con pywws* che descrive in dettaglio come installare pywws.

Se avete domande o non risposte nella documentazione, unitevi alla pywws Google mailing list / discussion group e chiedete lì. Si noti che il primo messaggio del gruppo non apparirà immediatamente - nuovi poster devono essere approvati da un moderatore, per evitare messaggi di spam.

3.1 Contenuto

3.1.1 Licenza Pubblica Generale GNU

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
     51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
```

```
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and


    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".
Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.
  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
```

```
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.
You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.
  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:
    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.
    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.
    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)
These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.
In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.
  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
```

```
Sections 1 and 2 above provided that you also do one of the following:
    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,
    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,
    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.
If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.
  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.
  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.
  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.
  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
```

```
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.
If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.
It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.
This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.
  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.
  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.
Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation.  If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.
  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission.  For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this.  Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.



    NO WARRANTY
  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
```

```
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.
  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.


     END OF TERMS AND CONDITIONS

    How to Apply These Terms to Your New Programs
  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.
  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:
    Gnomovision version 69, Copyright (C) year  name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.
You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.
  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice
This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
```

```
Public License instead of this License.
modification follow.
```

Commenti o domande? È possibile iscriversi alla mailing list pywws http://groups.google.com/group/pywws e farci sapere.

3.1.2 pywws Contributors

The copywright to pywws and its documentation is jointly held by the following contributors.

```
Developers
----------

Jim Easterbrook                 jim@jim-easterbrook.me.uk
x2q
3v1n0
Robin Kearney                   robin@kearney.co.uk
Rod Persky
Morten Høybye Frederiksen       morten@mfd-consult.dk
Simon Josefsson                 simon@josefsson.org
Matthew Hilton                  matthilton2005@gmail.com
Sabine Tobolka                                  oe1yvw@gmail.com
Markus Birth                    markus@birth-online.de
Chris Ramsay                    chris@ramsay-family.net
Christian Benke                 benkokakao@gmail.com
Ian Wilkinson                        null@sgtwilko.f9.co.uk

Translators
-----------

Edoardo                         edoardo69@hotmail.it
Jacques Desroches               metelsto@gmail.com
Sunshades                       joacim@ahlstrand.info
Johabu                          johabu96@yahoo.de
                                karte2@gmail.com
Kyle Gordon                     kyle@lodge.glasgownet.com>
Πτρο                           nouvakis@sch.gr
Ramiro                          ramiro.sanchez@telefonica.net
Rick Sulman                     rick@sulman.org
Pyttsen                         weather@spacelab.se
Tech2304                        tech2304@gmail.com
Pablo Vera                      pablo.vera82@gmail.com
```

Contributing to pywws

If you would like to add a feature to pywws (or fix a problem with it) then please do. Open source software thrives when its users become active contributors. The process is quite simple:

1. Join GitHub - it's free.
2. Fork the pywws repo - see Fork a Repo for help.
3. Clone your fork to a computer you can use to develop your new feature.

4. Use git to commit changes as you make them and push the changes to your fork of pywws.

   Please add a signed-off-by line to your commits which certify your developer certificate of origin (see below). For example, if your name is "John Smith", and your email address is «jsmith@example.com», just include the following line at the bottom of your commit messages:

   > Signed-off-by: John Smith <jsmith@example.com>

   You should be able to do this automatically by using the `-s` option on your `git commit` commands.

5. Add your name and email to the `src/contributors/contributors.txt` file. Don't forget the `-s` option when you commit this change.

6. Test your changes!

7. When everything's working as you expect, submit a Pull Request.

Developer Certificate of Origin

Including a signed-off-by line in your commits indicates that you certify the following:

```
Developer Certificate of Origin
Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.
660 York Street, Suite 102,
San Francisco, CA 94110 USA

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.


Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

Clauses (a), (b) and (c) reassure pywws users that the project will remain open source well in to the future. Clause (d) reminds you that your contributions will be publicly available, and you do not have the right to withdraw them in future.

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

3.1.3 Dipendenze

La lista di altri software da cui dipende pywws sembra spaventosamente lunga a prima vista. Tuttavia, molti di questi pacchetti non sono necessari nella maggior parte degli utenti. Che cosa avete bisogno dipende da cosa si vuole fare con pywws. Ricordate, è un «kit of parts» piuttosto che un'applicazione monolitica.

Some of the requirements are Python packages that can be downloaded from the Python Package Index (PyPI). I recommend using pip to install these.

You should be able to install the remaining dependencies using your operating system's package manager. This is a lot easier than downloading and compiling source files from the project websites. Note that some Linux distributions use different names for some of the packages, e.g. in Ubuntu, pyusb is called python-usb.

Nota: alcune di queste librerie possono avere le loro proprie dipendenze che potrebbe essere necessario installare. Segui i link per saperne di più su ciascuna libreria.

Indispensabile

- Python version 2.5 or higher

Python 3 è supportato, ma alcune cose potrebbero non funzionare correttamente. Se avete un problema con Python 3, si prega di inviare un messaggio al mailing list o presentare un <https://github.com/jim-easterbrook/pywws/issues>'_.

- pip

You will probably be able to install pip with your system's package manager, where it may be called python-pip or python3-pip or something similar. If not, download and run the `get-pip.py` file from the pip web site. In either case you should immediately use pip to install the latest version of itself:

```
sudo pip install --upgrade pip
```

Make sure you install the correct Python version's pip. If you want to install pywws for both Python 2 and Python 3 you will need pip2 and pip3.

- tzlocal

This is a handy little module that provides information on your local time zone. It's best installed with `pip`:

```
sudo pip install tzlocal
```

Libreria USB

Per recuperare dati da una stazione meteo pywws ha bisogno di una libreria che permette di comunicare via USB. C'è una varietà di librerie USB che possono essere utilizzate. Non tutte sono disponibili su tutte le piattaforme informatiche, che possono limitare la vostra scelta.

Mac OS X

On MacOS X the operating system's generic hid driver «claims» the weather station, which makes it very difficult to use any other USB interface. Unfortunately, you will need to download and compile hidapi yourself.

- hidapi
- ctypes (your package manager may know it as python-ctypes)

If you can't install ctypes then you can try the Cython interface to hidapi instead:

- cython-hidapi
- cython (your package manager may know it as python-Cython)

Other systems

Other systems use a Python interface to the libusb system library. There is a choice of interface and library version - install the latest that is available for your computer.

- libusb version 1.x (should be available from the package manager)
- python-libusb1 version 1.3

```
pip install libusb1
```

or

- libusb version 1.x or version 0.1 (should be available from the package manager)
- PyUSB version 1.0

```
pip install pyusb --pre
```

The `--pre` flag enables the installation of «pre release» versions, such as the current beta release (1.0.0b2) of pyusb.

If neither of these options works for you then you can use hidapi – see the Mac OS X instructions above.

Cambiato nella versione 15.01.0.dev1265: added ability to use python-libusb1 interface.

Flexible timed tasks

The `pywws.Tasks` module can do tasks at particular times and/or dates. This requires the croniter library. (Simple hourly, daily or "live" tasks don't need this library.)

- croniter

```
pip install croniter
```

Running as a daemon

The `pywws.livelogdaemon` program runs pywws live logging as a proper UNIX daemon process. It requires the python-daemon library:

- python-daemon

```
pip install python-daemon
```

Disegnare grafici

The `pywws.Plot` module uses gnuplot to draw graphs. If you want to produce graphs of weather data, e.g. to include in a web page, you need to install the gnuplot application:

- gnuplot v4.2 or higher (should be available from the package manager)

After installing gnuplot you should edit weather.ini (see *weather.ini - configurazione del formato del file*) and set the `gnuplot version` config item. Finding out the installed gnuplot version is easy:

```
gnuplot -V
```

Trasferimento sicuro di file (sftp)

Il modulo *pywws.Upload* può utilizzare «ftp over ssh» (sftp) per caricare i file sul vostro sito web. Il caricamento normale utilizza i moduli Python standard, ma se si desidera utilizzare sftp è necessario installare questi due moduli:

- paramiko
- pycrypto

```
sudo pip install pycrypto paramiko
```

Postare su Twitter

Il modulo *pywws.ToTwitter* è utilizzato per inviare messaggi delle condizioni meteo a Twitter. Per postare su Twitter richiede questi moduli:

- python-twitter v3.0 or higher
- python-oauth2

```
sudo pip install python-twitter oauth2
```

or

- tweepy v2.0 or higher
- python-oauth2

```
sudo pip install tweepy oauth2
```

Note that `tweepy` appears to be the less reliable of the two. If you have problems, e.g. with character encoding, try installing `python-twitter` instead.

Cambiato nella versione 13.10_r1086: Riabilitato uso della libreria `tweepy` come un'alternativa a `python-twitter`. `python-oauth2` è ancora richiesto da *pywws.TwitterAuth*.

Cambiato nella versione 13.06_r1023: Pywws precedentemente utilizzava la libreria `tweepy` invece di `python-twitter` e `python-oauth2`.

MQTT

Nuovo nella versione 14.12.0.dev1260.

The `pywws.toservice` module can be used to send weather data to an MQTT broker. This requires the paho-mqtt module:

- paho-mqtt

```
sudo pip install paho-mqtt
```

Per creare nuove traduzioni di lingua

pywws can be configured to use languages other than English, as described in *Come utilizzare pywws in un altra lingua*. The Babel package is required to extract the strings to be translated and to compile the translation files.

- babel

```
sudo pip install babel
```

Copying files to or from Transifex requires the transifex-client package.

- transifex-client

```
sudo pip install transifex-client
```

Translating the documentation using local files needs the sphinx-intl package.

- sphinx-intl

```
sudo pip install sphinx-intl
```

Cambiato nella versione 14.05.dev1209: pywws previously used the gettext package.

Per "compilare" la documentazione

The documentation of pywws is written in «ReStructured text». A program called Sphinx is used to convert this easy to write format into HTML for use with a web browser. If you'd like to create a local copy of the documentation (so you don't have to rely on the online version, or to test a translation you're working on) you need to install Sphinx, version 1.3 or later.

- Sphinx

```
sudo pip install sphinx
```

Commenti o domande? È possibile iscriversi alla mailing list pywws http://groups.google.com/group/pywws e farci sapere.

3.1.4 Storico aggiornamenti

```
pywws - Python software for USB Wireless Weather Stations
http://github.com/jim-easterbrook/pywws
Copyright (C) 2008-18  pywws contributors

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
```

```
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.

Changes in v18.04.0:
    1/ Now works with broken weather stations that have stopped logging data.
       (Although there may be more gaps in the data.)
    2/ Cope better with missing wind direction data.
    3/ Use HTTPS for Weather Underground uploads.
    4/ Other minor bug fixes.

Changes in v17.11.0:
    1/ Allow MQTT service without password.
    2/ Allow SFTP uploads with public key.
    3/ Increase Twitter character limit from 140 to 280.
    4/ Various other bug fixes and minor improvements.

Changes in v16.12.0:
    1/ Added "candlestick" plot type.
    2/ Added cloud base calculation function.
    3/ Various other bug fixes and minor improvements.

Changes in v16.08.0:
    1/ Fix Python 2.5 incompatibilities.
    2/ Fix python-twitter v3 tweet length problem.

Changes in v16.07.1:
    1/ Further changes to handle UK Met Office server quirks.

Changes in v16.07.0:
    1/ Fix bug with UK Met Office uploads server change.
    2/ Allow user commands in wind roses.
    3/ Various other bug fixes and minor improvements.

Changes in v15.12.0:
    1/ Fix bug with Twitter messages being excessively truncated.
    2/ Improve handling of utf-8 encoded templates.
    3/ Improved plots and wind roses with 'pngcairo' "terminal".
    4/ Various bug fixes and minor improvements.

Changes in v15.11.0:
    1/ Add Russian translation of program text.
    2/ Improved documentation.
    3/ Various bug fixes and minor improvements.

Changes in v15.07.0:
    1/ Can include multiple media in Twitter messages.
    2/ Attempt to fix bug in wind rose axes labels.
    3/ Enable inclusion of time & date in wind rose title.
    4/ Various bug fixes and minor improvements.

Changes in v15.01.0:
    1/ Added 'MQTT' service.
    2/ Added another USB library option.
```

```
    3/ Improved Python 3 compatibility.
    4/ Various bug fixes and minor improvements.

Changes in v14.12.0:
    1/ Updated temperatur.nu and wetterarchiv.de service details to suit new
       APIs.

Changes in v14.06.1:
    1/ Revised version numbering scheme.
    2/ Compiled documentation no longer included in releases.
    3/ Can partially specify start & stop date/time in graphs, e.g. to start a
       plot at midnight, no matter when it is plotted.

Changes in v14.06:
    1/ Can now send images to Twitter.
    2/ Periodic tasks can be specified with a cron style syntax.
    3/ Added wind direction filter for use in graphs or user calibration
       modules.
    4/ Wind direction is now stored as a float. Old templates that use the
       wind_dir_text array will need updating, probably to use the
       winddir_text() function.
    5/ Started using "Transifex" to host translations. Changed tools and
       procedures to create new translations.
    6/ Improved USB hangup avoidance strategy for stations with large clock
       drift figures.
    7/ Various bug fixes and minor improvements.

Changes in v14.05:
    1/ Rearranged package layout, moving examples and documentation.
    2/ Added 'entry point' auto-generated commands for some modules.
    3/ Added verbose output option to pywws-version command.
    4/ Various bug fixes and minor improvements.

Changes in v14.03:
    1/ Extracts additional status from 'wind_dir' byte. You must run
       pywws-reprocess.py with the -u option after upgrading from any previous
       version.
    2/ Added Citizen Weather Observer Program to available 'services'.
    3/ Improved asynchronous upload task queuing.
    4/ Various bug fixes and minor improvements.

Changes in v14.02:
    1/ Improved time zone handling, including non whole hour time zones.
    2/ New 'frequent writes' config option.
    3/ Improved 'live log' sync, particularly with 3080 type stations.
    4/ Record recent memory pointer to improve detection of gaps in data.
    5/ Various bug fixes and minor improvements.

Changes in v13.12:
    1/ Changed API of user calibration module.
    2/ Can use python-twitter *or* tweepy library.
    3/ Added a script to run live logging as a UNIX daemon process.
    4/ Changed data store to use separate read and write caches.
    5/ Various bug fixes and minor improvements.

Changes in v13.10:
    1/ Changed Twitter library from tweepy to python-twitter.
    2/ Added ability to do uploads asynchronously.
```

```
    3/ Rearranged and improved documentation.
    4/ Various bug fixes and minor improvements.

Changes in v13.06:
    1/ Substantially rearranged directories, getting rid of 'code' and 'code3'.
    2/ Removed 'makefile' - everything is now done via 'setup.py'.
    3/ Removed 'RunModule.py' - use 'python -m pywws.module' now.
    4/ Separated storage of config (weather.ini) and status (status.ini).
    5/ Replaced toservice.py "rapid fire" mode with a separate config file for
       Weather Underground rapid fire.
    6/ Added 2 more low-level USB access modules.
    7/ Various bug fixes and minor improvements.

Changes in v13.03:
    1/ Added 'rain days' to monthly data. (Reprocess required when upgrading.)
    2/ Extended template syntax to include comments.
    3/ Added 'humidity index' function.
    4/ Added French translation of documentation.
    5/ Reduced frequency of saving data files.
    6/ Various bug fixes.

Changes in v12.12:
    1/ Added support for Python 3.
    2/ Added French documentation translation.
    3/ Used 'binary search' to speed up data access.
    4/ Various bug fixes.

Changes in v12.11:
    1/ Moved development from Google code to GitHub.
    2/ Made software attempt to avoid USB activity at times when it is assumed
       the weather station might be writing to its memory. This might solve
       the USB lockup problem, but it's too early to tell.

Changes in v12.10:
    1/ Added a 'winddir_text' function for use in templates.
    2/ Added <ytics> and <y2tics> options to graph plots.
    3/ Various bug fixes.

Changes in v12.07:
    1/ Added Open Weather Map to the services.
    2/ Fixed problem with Weather Underground uploads that started on 1st June.
    3/ Various bug fixes and software structure improvements.

Changes in v12.05:
    1/ Made 'fixed block' data available to template calculations.
    2/ Fixed buggy auto-detection of 3080 weather stations.
    3/ Added a function to generate the Zambretti forecast code letter.
    4/ Added a program to test USB communication reliablility.
    5/ Various bug fixes and software structure improvements.

Changes in v12.02:
    1/ Separated out low level USB communications to enable use of different
       libraries. Now works on recent versions of Mac OS.
    2/ Added humidity, pressure & wind data to summary data.
    3/ Merged Weather Underground and UK Met Office uploaders into one combined
       module. Added more 'service' uploaders.
    4/ Various bug fixes and software structure improvements.
```

```
Changes in v11.10:
    1/ Complete restructuring of documentation.
    2/ Added a user defined 'calibration' process.
    3/ Sets 'locale' according to language setting.
    4/ Added ability to upload to UK Met Office 'WOW'.
    5/ Various bug fixes and software structure improvements.
    6/ New language files: French, Danish.

Changes in v11.05:
    1/ Added support for '3080' family stations that have illuminance and
       UV sensors.
    2/ Broadened the range of tasks that can be done with 'live' data.
    3/ Various bug fixes and software structure improvements.

Changes in v11.02:
    1/ Various bug fixes and software structure improvements.
    2/ Improved wind direction averaging.
    3/ Added conversion functions for common things such as C to F.
    4/ Added a YoWindow module.
    5/ Improved Zambretti forecaster.

Changes in v10.12:
    1/ Various bug fixes and software structure improvements.
    2/ Added a 'goto' instruction to Template.py.
    3/ Added a 'Zambretti' forecast function to Template.py. This should
       be treated as an experiment, and not relied upon for accuracy.

Changes in v10.10:
    1/ Added 'catchup' mode to ToUnderground.py.
    2/ Created 'Tasks.py' to handle common tasks.
    3/ Made better use of Python's logger for info and error
       messages.
    4/ Changed over from 'python-twitter' to 'tweepy' for Twitter
       access. Twitter authorisation using OAuth now works.
    5/ Added 'LiveLog.py' live logging program.
    6/ Added 'SetWeatherStation.py' to do some configuration of weather
       station. No longer need EasyWeather to set logging interval!
    7/ Added 'Rapid Fire' ability to ToUnderground.py.
    8/ Added plain text versions of HTML documentation.
    9/ Many bug fixes and minor improvements.

Changes in v10.08:
    1/ Added internal temperature to daily and monthly summaries.
       Run Reprocess.py when upgrading from earlier versions.
    2/ Added 'prevdata' to Template.py. Allows calculations that
       compare values from different times.
    3/ Made 'pressure_offset' available to calculations in Plot.py
       and Template.py. This is only useful when using 'raw' data.
    4/ Improved synchronisation to weather station's clock when
       fetching stored data.

Changes in v10.06:
    1/ Improved localisation code.
    2/ Minor bug fixes.
    3/ Added Y axis label angle control to plots.

Changes in v10.04:
    1/ Changed version numbering to year.month.
```

```
    2/ Allowed "upload" to a local directory instead of ftp site.
    3/ Added "calc" option to text templates (Template.py).
    4/ Added -v / --verbose option to Hourly.py to allow silent operation.
    5/ Added internationalisation / localisation of some strings.
    6/ Made 'raw' data available to text templates.
    7/ Added ability to upload to Weather Underground.
    8/ Added dual axis and cumulative graph capability.

Changes in v0.9:
    1/ Added lowest daytime max and highest nighttime min temperatures
       to monthly data.
    2/ Added average temperature to daily and monthly data.
    3/ Added 'terminal' element to Plot.py templates for greater control
       over output appearance.
    4/ Added 'command' element to Plot.py templates for even more
       control, for advanced users.
    5/ Added secure upload option.
    6/ Minor speed improvements.

Changes in v0.8:
    1/ Added meteorological day end hour user preference
    2/ Attempts at Windows compatibility
    3/ Corrected decoding of wind data at speeds over 25.5 m/s
    4/ Improved speed with new data caching strategy

Changes in v0.7:
    1/ Several bug fixes, mostly around new weather stations with not
       much data
    2/ Added min & max temperature extremes to monthly data
    3/ Added template and workspace directory locations to weather.ini
    4/ Increased versatility of Plot.py with layout and title elements

Changes in v0.6:
    1/ Added monthly data
    2/ Changed 'pressure' to 'abs_pressure' or 'rel_pressure'

Changes in v0.5:
    1/ Small bug fixes.
    2/ Added start time to daily data
    3/ Replaced individual plot programs with XML "recipe" system

Changes in v0.4:
    1/ Can post brief messages to Twitter.
    2/ Now time zone aware. Uses UTC for data indexing and local time
       for graphs and text data files.

Changes in v0.3:
    1/ Now uses templates to generate text data
    2/ Added 28 day plot
    3/ Minor efficiency improvements
    4/ Improved documentation

Changes in v0.2:
    1/ Now uses Python csv library to read and write data
    2/ Creates hourly and daily summary files
    3/ Includes rain data in graphs
```

Commenti o domande? È possibile iscriversi alla mailing list pywws http://groups.google.com/group/pywws e farci sapere.

3.1.5 Guide per gli utenti

Contenuti:

Come iniziare con pywws

Installation

First of all you need to install Python and a USB library (to allow Python to access the weather station). See *Dipendenze* for more detail.

Creare una directory per tutti i file meteo correlati e posizionarsi nella directory. Per esempio (su un sistema Linux o simile sistema operativo):

```
mkdir ~/weather
cd ~/weather
```

Installazione facile

The easiest way to install pywws is with the pip command:

```
sudo pip install pywws
```

Aggiornare pywws con una semplice riga di comando:

```
sudo pip install -U pywws
```

Now you are ready to *Testare la connessione della stazione meteo*.

Scaricare ed estrarre

If you prefer not to use pip, or you want easy access to the pywws source files (e.g. to translate the documentation – see *Come utilizzare pywws in un altra lingua*), you can download and extract the files into your weather directory.

Visit http://pypi.python.org/pypi/pywws/ and download one of the .tar.gz or .zip files. Put it in your weather directory, then extract all the files, for example:

```
cd ~/weather
tar zxvf pywws-14.03.dev1178.tar.gz
```

o:

```
cd ~/weather
unzip pywws-14.03.dev1178.zip
```

This should create a directory (called `pywws-14.03.dev1178` in this example) containing all the pywws source files. It is convenient to create a soft link to this awkwardly named directory:

```
cd ~/weather
ln -s pywws-14.03.dev1178 pywws
```

L'aggiornamento di una versione scaricata è lo stesso processo come per la prima installazione. Scarica il .tar.gz o .zip file, estrai il suo contenuto, quindi elimina il link che punta al vecchio download e creane uno che punta al nuovo download. Una volta che siete soddisfatti e la nuova versione funziona OK, è possibile eliminare il vecchio download interamente.

Clone the repository

The PyPI files contain a snapshot release of the software - a new one is issued every few months. If you want to use the very latest version of pywws, e.g. to work on fixing a bug, you can get all the files you need from the GitHub repository. Install git and use it to clone the repos:

```
cd ~/weather
git clone https://github.com/jim-easterbrook/pywws.git
```

To upgrade you use git to pull any changes:

```
cd ~/weather/pywws
git pull
```

Install pywws

If you have downloaded or cloned the pywws source files, you need to use setup.py to install it:

```
cd ~/weather/pywws
python setup.py compile_catalog
python setup.py build
sudo python setup.py install
```

The `python setup.py compile_catalog` step is only needed if you want to use pywws in a language other than English. See *Test the pywws translations* for more detail.

Note to Python 3 users: this will generate and use Python 3 versions of the pywws software in `~/weather/pywws/build/lib`.

Compile documentation (optional)

If you'd like to have a local copy of the pywws documentation (and have downloaded the source or cloned the repo) you can «compile» the English documentation. This requires the sphinx package:

```
cd ~/weather/pywws
python setup.py build_sphinx
```

Compiling the documentation in another language requires the additional step of compiling the translation files, which requires the sphinx-intl package. For example, to compile the French documentation:

```
cd ~/weather/pywws
sphinx-intl build --locale-dir src/pywws/lang -l fr
LANG=fr python setup.py build_sphinx
```

The compiled documentation should then be found at `~/weather/pywws/doc/html/index.html`. See *Come utilizzare pywws in un altra lingua* for more detail.

Testare la connessione della stazione meteo

Now you're ready to test your pywws installation. Connect the weather station (if not already connected) then run the *pywws.TestWeatherStation* module:

```
pywws-testweatherstation
```

Se tutto funziona correttamente, questo dovrebbe visualizzare dei numeri sullo schermo, ad esempio:

```
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00␣
↪0f 00 00 60 55
0020 ea 27 a0 27 00 00 00 00 00 00 00 10 10 12 13 45 41 23 c8 00 32 80 47 2d 2c 01 2c␣
↪81 5e 01 1e 80
0040 96 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 03 00 00␣
↪00 00 00 00 00
0060 00 00 4e 1c 63 0d 2f 01 73 00 7a 01 47 80 7a 01 47 80 e4 00 00 00 71 28 7f 25 bb␣
↪28 bd 25 eb 00
0080 0c 02 84 00 0e 01 e3 01 ab 03 dc 17 00 10 08 21 08 54 10 03 07 22 18 10 08 11 08␣
↪30 10 04 21 16
00a0 26 08 07 24 17 17 08 11 01 06 10 09 06 30 14 29 09 01 06 07 46 09 06 30 14 29 09␣
↪01 06 07 46 08
00c0 08 31 14 30 10 05 14 15 27 10 01 26 20 47 09 01 23 05 13 10 01 26 20 47 09 01 23␣
↪05 13 10 02 22
00e0 11 06 10 02 22 11 06 08 07 07 19 32 08 12 13 22 32 08 09 07 08 48 01 12 05 04 43␣
↪10 02 22 14 43
```

Ci sono diverse ragioni perché questo potrebbe non funzionare. Molto probabilmente è un problema di "permessi". Questo può essere testato eseguendo il comando come root:

```
sudo pywws-testweatherstation
```

If this works then you may be able to allow your normal user account to access the weather station by setting up a "udev" rule. The exact method may depend on your Linux version, but this is typically done by creating a file `/etc/udev/rules.d/39-weather-station.rules` containing the following:

```
ACTION!="add|change", GOTO="weatherstation_end"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1941", ATTRS{idProduct}=="8021", GROUP=
↪"weatherstation"
LABEL="weatherstation_end"
```

Unplug and replug the station's USB connection to force `udev` to apply the new rule. This allows any user in the group `weatherstation` to access the weather station. You need to create this group and add your normal user account to it – many Linux systems have a GUI for user and group management.

Se avete qualsiasi altro problema, per favore chiedi aiuto pywws mailing list: http://groups.google.com/group/pywws

Configurare la tua stazione meteo

Se non lo hai già fatto, imposta la tua stazione meteo per visualizzare la corretta pressione atmosferica relativa. (Vedere il manuale per informazioni su come effettuare questa operazione). pywws ottiene l'offset tra pressione relativa e assoluta dalla stazione, quindi questa deve essere impostata prima di utilizzare pywws.

Cercando su internet tra i bollettini meteo di una stazione vicina, idealmente una ufficiale come un aeroporto, è possibile ottenere la corretta pressione relativa dalla vostra posizione. Questo è meglio farlo quando il tempo è calmo e la pressione è costante su una vasta area.

Impostare l'intervallo di registrazione della stazione meteo

La tua stazione meteo probabilmente lasciato la fabbrica con un intervallo di registrazione di 30 minuti. In questo modo la stazione memorizzare circa 11 settimane di dati. La maggior parte degli utenti pywws usano i loro computer per leggere i dati dalla stazione ogni ora, o più spesso che solo bisogno la stazione per archivi dati sufficienti per coprire i fallimenti del computer. L'intervallo consigliato è di 5 minuti, che consente ancora 2 settimane di stoccaggio. Per impostare l'intervallo usa *pywws.SetWeatherStation*:

```
pywws-setweatherstation -r 5
```

Note that the weather station will not start using the new interval until the current 30 minute logging period is finished. This may cause «station is not logging data» errors when running pywws logging. If this happens you need to wait until the 30 minute logging period ends.

Registrare i dati della stazione meteo

Innanzitutto, scegliere una directory per archiviare tutti i dati della stazione meteo. Questo verrà scritto abbastanza frequentemente, quindi un disco rigido è preferibile a una memory stick, come questi hanno un numero limitato di scritture. Nella maggior parte dei casi è adatta la home directory, per esempio:

```
mkdir ~/weather/data
```

Questa directory viene definita nella documentazione pywws altrove come directory di dati.

Assicurarsi che il computer abbia la giusta data ora e fuso orario, poiché questi vengono usati per etichettare i dati della stazione meteo. Se non hai già fatto, vale la pena di impostazione NTP per sincronizzare il computer a un "time server".

La prima volta che si esegue *pywws.LogData* sarà creato un file di configurazione nella directory dati chiamato "weather.ini" e poi chiudi il programma. È necessario modificare il file di configurazione e cambiare la linea `ws type = Unknown` in `ws type = 1080` o `ws type = 3080`. (Se la tua stazione meteo visualizza illuminamento solare avete un modello 3080, tutti gli altri sono 1080). Quindi eseguire nuovamente *pywws.LogData*. Ciò può richiedere diversi minuti, siccome esso copierà tutti i dati memorizzati nella memoria della vostra stazione. Il programma *pywws.LogData* ha un'opzione "verbose" che aumenta la quantità di messaggi vengono visualizzati durante l'esecuzione. Questo è utile quando si esegue manualmente, ad esempio:

```
python -m pywws.LogData -vvv ~/weather/data
```

(Sostituire `~/weather/data` con la directory dei dati, se è diversa.)

Ora dovreste avere alcuni file di dati che si possono guardare. Ad esempio:

```
more ~/weather/data/raw/2012/2012-12/2012-12-16.txt
```

(Sostituire l'anno, il mese e il giorno con quelli che i vostri dati dovrebbero avere.)

Convertire i vecchi dati di EasyWeather (opzionale)

Se era in esecuzione EasyWeather prima di decidere di utilizzare pywws, è possibile convertire i dati che EasyWeather aveva registrato nel formato pywws. Trovare il file EasyWeather.dat e poi convertirlo:

```
python -m pywws.EWtoPy EasyWeather.dat ~/weather/data
```

Impostare alcune opzioni di configurazione

Dopo l'esecuzione *pywws.LogData* ci dovrebbe essere un file di configurazione nella directory dati chiamata "weather.ini". Aprire questa con un editor di testo. Si dovrebbe trovare qualcosa come il seguente:

```
[config]
ws type = 1080
logdata sync = 1
pressure offset = 9.4
```

È necessario aggiungere una nuova voce nella sezione `[config]` chiamata `day end hour`. Questo dice a pywws che convenzione si desidera utilizzare nel calcolo dei dati di riepilogo giornalieri. Nel Regno Unito, la "giornata meteorologica" è solitamente dalle 09.00 alle 09:00 GMT (10.00 alle 10.00 BST durante l'estate), quindi utilizzare un valore di ora di fine giornata di 9. In altri paesi il valore 24 (o 0) potrebbe essere più adatto. Si noti che il valore è impostato nel periodo invernale locale. Non è necessario cambiarla quando l'ora legale è in vigore.

Dopo la modifica, il file weather.ini dovrebbe apparire qualcosa di simile a questo:

```
[config]
ws type = 1080
logdata sync = 1
pressure offset = 9.4
day end hour = 9
```

È inoltre possibile modificare il valore `pressure offset` per regolare il calcolo di pywws della pressione relativa (al livello del mare) dal valore assoluto della stazione. In futuro se si modifica la pressione offset o l'ora di fine giornata, è necessario aggiornare tutti i dati memorizzati tramite il comando `pywws.Reprocess`.

Per maggiori dettagli sulle opzioni del file di configurazione, vedere *weather.ini - configurazione del formato del file*.

Cambiato nella versione 13.10_r1082: inserito `pressure offset` un elemento di configurazione. In precedenza è sempre stata letta dalla stazione meteo.

Elaborare i dati grezzi(raw)

pywws.LogData copia solo i dati grezzi dalla stazione meteo. Per fare qualcosa di utile con quei dati è probabilmente necessario di riepiloghi ogni ora, giornalieri e mensili. Sono creati da `pywws.Process`. Ad esempio:

```
python -m pywws.Process ~/weather/data
```

Ora dovreste avere alcuni file elaborati da guardare:

```
more ~/weather/data/daily/2012/2012-12-16.txt
```

Se cambi l'impostazione di configurazione `day end hour`, è necessario rielaborare tutti i dati meteo. È possibile farlo eseguendo `pywws.Reprocess`:

```
python -m pywws.Reprocess ~/weather/data
```

Ora siete pronti per impostare la registrazione ad intervalli o continua, come descritto in *Come impostare "hourly" per la registrazione oraria con pywws* o *Come impostare una registrazione "live" con pywws*.

Leggere la documentazione

You're looking at it right now! The *Guide per gli utenti* section is probably the most useful bit to read first, but the *Programmi Python e moduli* section has a lot more detail on the various pywws modules and commands.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come impostare "hourly" per la registrazione oraria con pywws

Introduzione

There are two quite different modes of operation with pywws. Traditionally `pywws.Hourly` would be run at regular intervals (usually an hour) from cron. This is suitable for fairly static websites, but more frequent updates can be useful for sites such as Weather Underground (http://www.wunderground.com/). The newer `pywws.LiveLog` program runs continuously and can upload data every 48 seconds.

Note that although this document (and the program name) refers to "hourly" logging, you can run `pywws.Hourly` as often or as infrequently as you like, but don't try to run it more often than double your logging interval. For example, if your logging interval is 10 minutes, don't run `pywws.Hourly` more often than every 20 minutes.

Guida introduttiva

Prima di tutto, è necessario installare pywws e assicurarsi che si possono ottenere i dati dalla tua stazione meteo. Vedere *Come iniziare con pywws* per i dettagli.

Try running `pywws.Hourly` from the command line, with a high level of verbosity so you can see what's happening. Use the `pywws-hourly` command to run `pywws.Hourly`:

```
pywws-hourly -vvv ~/weather/data
```

Entro cinque minuti (supponendo di aver impostato un intervallo di registrazione di 5 minuti) si dovrebbe vedere un messaggio "live_data new ptr", seguita dal recupero di eventuali nuovi dati dalla stazione meteo ed elaborarli.

Cambiato nella versione 14.04.dev1194: the `pywws-hourly` command replaced `scripts/pywws-hourly.py`.

Percorsi dei file di configurazione

Aprire il file weather.ini con un editor di testo. Si dovrebbe avere una sezione di `[paths]` simile al seguente (dove `xxx` è il tuo nome utente):

```
[paths]
work = /tmp/weather
templates = /home/xxx/weather/templates/
graph_templates = /home/xxx/weather/graph_templates/
local_files = /home/xxx/weather/results/
```

Edit these to suit your installation and preferences. `work` is an existing temporary directory used to store intermediate files, `templates` is the directory where you keep your text template files, `graph_templates` is the directory where you keep your graph template files and `local_files` is a directory where template output that is not uploaded to your web site is put. Don't use the pywws example directories for your templates, as they will get over-written when you upgrade pywws.

Copy your text and graph templates to the appropriate directories. You may find some of the examples provided with pywws useful to get started. The `pywws-version -v` command should show you where the examples are on your computer.

Nuovo nella versione 14.04.dev1194: the `pywws-version` command.

Configurazione dell'esecuzione periodica

In weather.ini si dovrebbe avere le sezioni `[logged]`, `[hourly]`, `[12 hourly]` e `[daily]` simili ai seguenti:

```
[logged]
services = []
plot = []
text = []

[hourly]
...
```

These specify what `pywws.Hourly` should do when it is run. Tasks in the `[logged]` section are done every time there is new logged data, tasks in the `[hourly]` section are done every hour, tasks in the `[12 hourly]` section are done twice daily and tasks in the `[daily]` section are done once per day.

La voce `services` è un elenco di servizi meteo online per inviare i dati meteo. Le voci `plot` e `text` sono elenchi di file di modello grafico e file di testo per essere elaborati e, opzionalmente, caricati su un sito web. Aggiungere i nomi dei file di modello e servizi meteo alle voci appropriate, ad esempio:

```
[logged]
services = ['underground', 'metoffice']
plot = []
text = []

[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_24hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']

[12 hourly]
services = []
plot = []
text = []

[daily]
services = []
plot = ['28days.png.xml']
text = [('forecast.txt', 'T'), 'allmonths.txt']
```

Si noti l'uso del flag `'T'` – questo dice a pywws si inviare il risultato a Twitter invece di caricarlo sul proprio sito ftp.

You can test that all these are working by removing the `[last update]` section from status.ini, then running `pywws.Hourly` again:

```
pywws-hourly -v ~/weather/data
```

Nuovo nella versione 14.05.dev1211: `[cron name]` sections. If you need more flexibility in when tasks are done you can use `[cron name]` sections. See *weather.ini - configurazione del formato del file* for more detail.

Cambiato nella versione 13.06_r1015: Aggiunto il flag `'T'`. Precedentemente i modelli Twitter sono stati indicati separatamente in voci `twitter` in `[hourly]` e di altre sezioni. La sintassi precedente funziona ancora, ma è obsoleta.

Cambiato nella versione 13.05_r1009: the last update information was previously stored in weather.ini, with `last update` entries in several sections.

Eseguire con cron job

Most UNIX/Linux systems have a "cron" daemon that can run programs at certain times, even if you are not logged in to the computer. You edit a "crontab" file to specify what to run and when to run it. For example, to run `pywws.Hourly` every hour, at zero minutes past the hour:

```
0 * * * *       pywws-hourly /home/xxx/weather/data
```

This might work, but if it didn't you probably won't get any error messages to tell you what went wrong. It's much better to run a script that runs `pywws.Hourly` and then emails you any output it produces. Here's the script I use:

```
#!/bin/sh
#
# weather station logger calling script

export PATH=$PATH:/usr/local/bin

if [ ! -d ~/weather/data/ ]; then
  exit
  fi

log=/var/log/log-weather

pywws-hourly -v ~/weather/data >$log 2>&1

# mail the log file
/home/jim/scripts/email-log.sh $log "weather log"
```

Sarà necessario modificare questo file un bel pò per soddisfare i vostri percorsi di file e così via, ma dà un'idea di cosa fare.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come impostare una registrazione "live" con pywws

Introduzione

There are two quite different modes of operation with pywws. Traditionally `pywws.Hourly` would be run at regular intervals (usually an hour) from cron. This is suitable for fairly static websites, but more frequent updates can be useful for sites such as Weather Underground (http://www.wunderground.com/). The newer `pywws.LiveLog` program runs continuously and can upload data every 48 seconds.

Guida introduttiva

Prima di tutto, è necessario installare pywws e assicurarsi che si possono ottenere i dati dalla tua stazione meteo. Vedere *Come iniziare con pywws* per i dettagli.

If you have previously been using `pywws.Hourly` then disable your "cron" job (or whatever else you use to run it) so it no longer runs. You should not run `pywws.Hourly` and `pywws.LiveLog` at the same time.

Try running `pywws.LiveLog` from the command line, with a high level of verbosity so you can see what's happening. Use the `pywws-livelog` command to run `pywws.LiveLog`:

```
pywws-livelog -vvv ~/weather/data
```

Within five minutes (assuming you have set a 5 minute logging interval) you should see a "live_data new ptr" message, followed by fetching any new data from the weather station and processing it. Let `pywws.LiveLog` run for a minute or two longer, then kill the process by typing "<Ctrl>C".

Cambiato nella versione 14.04.dev1194: the `pywws-livelog` command replaced `scripts/pywws-livelog.py`.

Percorsi dei file di configurazione

Aprire il file weather.ini con un editor di testo. Si dovrebbe avere una sezione di `[paths]` simile al seguente (dove `xxx` è il tuo nome utente):

```
[paths]
work = /tmp/weather
templates = /home/xxx/weather/templates/
graph_templates = /home/xxx/weather/graph_templates/
local_files = /home/xxx/weather/results/
```

Edit these to suit your installation and preferences. `work` is an existing temporary directory used to store intermediate files, `templates` is the directory where you keep your text template files, `graph_templates` is the directory where you keep your graph template files and `local_files` is a directory where template output that is not uploaded to your web site is put. Don't use the pywws example directories for your templates, as they will get over-written when you upgrade pywws.

Copy your text and graph templates to the appropriate directories. You may find some of the examples provided with pywws useful to get started. The `pywws-version -v` command should show you where the examples are on your computer.

Nuovo nella versione 14.04.dev1194: the `pywws-version` command.

Configurazione dell'esecuzione periodica

In weather.ini si dovrebbe avere una sezione `[live]` simile alla seguente:

```
[live]
services = []
plot = []
text = []
```

Questa sezione specifica cosa pywws dovrebbe fare ogni volta che ottiene una nuova lettura dalla stazione meteo, cioè ogni 48 secondi. La voce `services` è un elenco di servizi meteo online dove caricare dati, ad esempio

['underground_rf']. Le voci plot e text sono elenchi di file di modello grafici e file di testo che devono essere elaborati e, opzionalmente, caricati sul tuo sito web. Probabilmente si dovrebbe lasciare tutto questo vuoto tranne che per i servizi services.

Se si utilizza YoWindow (http://yowindow.com/) si può aggiungere una voce alla sezione [live] per specificare il file di YoWindow, ad esempio:

```
[live]
services = ['underground_rf']
text = [('yowindow.xml', 'L')]
...
```

Si noti l'uso del flag 'L' – questo dice a pywws si inviare il risultato alla directory «local files» invece di caricarlo sul tuo sito ftp.

Se non li hai già, create quattro altre sezioni nel file weather.ini: [logged], [hourly], [12 hourly] e [daily]. Queste sezioni dovrebbero avere voci simili alla sezione [live] e specificare cosa fare ogni volta i dati vengono registrati (5-30 minuti, a seconda l'intervallo di registrazione), ogni ora, due volte al giorno e una volta al giorno. Aggiungere i nomi dei file di modello alle voci appropriate, ad esempio:

```
[logged]
services = ['underground', 'metoffice']
plot = []
text = []

[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_24hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']

[12 hourly]
services = []
plot = []
text = []

[daily]
services = []
plot = ['28days.png.xml']
text = [('forecast.txt', 'T'), 'allmonths.txt']
```

Si noti l'uso del flag 'T' – questo dice a pywws si inviare il risultato a Twitter invece di caricarlo sul proprio sito ftp.

Nuovo nella versione 14.05.dev1211: [cron name] sections. If you need more flexibility in when tasks are done you can use [cron name] sections. See *weather.ini - configurazione del formato del file* for more detail.

Cambiato nella versione 13.06_r1015: Aggiunto il flag 'T'. Precedentemente i modelli Twitter sono stati indicati separatamente in voci twitter in [hourly] e di altre sezioni. La sintassi precedente funziona ancora, ma è obsoleta.

Cambiato nella versione 13.05_r1013: aggiunto il modello 'yowindow.xml'. Precedentemente il file yowindow era generato da un modulo separato, richiamato dalla voce yowindow nella sezione [live]. La sintassi precedente funziona ancora, ma è obsoleta.

Upload asincrono

Nuovo nella versione 13.09_r1057.

Il caricamento dei dati in siti web o "services" a volte può richiedere del tempo, in particolare se un sito è andato off line e i tempi di caricamento sono lunghi. In condizioni di normale funzionamento pywws attende che tutti i caricamenti siano elaborati prima di scaricare ulteriori dati dalla stazione meteo. Questo può portare alcune volte a dati di essere mancanti.

The `asynchronous` item in the `[config]` section of weather.ini can be set to `True` to tell `pywws.LiveLog` to do these uploads in a separate thread.

Esecuzione in background

Nuovo nella versione 13.12.dev1118.

In order to have `pywws.LiveLog` carry on running after you finish using your computer it needs to be run as a «background job». On most Linux / UNIX systems you can do this by putting an ampersand ("&") at the end of the command line. Running a job in the background like this doesn't always work as expected: the job may suspend when you log out. It's much better to run as a proper UNIX "daemon" process.

The `pywws.livelogdaemon` program does this, if you have the python-daemon library installed:

```
pywws-livelog-daemon -v ~/weather/data ~/weather/data/pywws.log start
```

Note that the log file is a required parameter, not an option.

Riavvio automatico

There are various ways of configuring a Linux system to start a program when the machine boots up. Typically these involve putting a file in `/etc/init.d/`, which requires root privileges. A slightly harder problem is ensuring a program restarts if it crashes. My solution to both problems is to run the following script from cron, several times an hour.

```
#!/bin/sh

export PATH=$PATH:/usr/local/bin

# exit if NTP hasn't set computer clock
[ `ntpdc -c sysinfo | awk '/stratum:/ {print $2}'` -ge 10 ] && exit

pidfile=/var/run/pywws.pid
datadir=/home/jim/weather/data
logfile=$datadir/live_logger.log

# exit if process is running
[ -f $pidfile ] && kill -0 `cat $pidfile` && exit

# email last few lines of the logfile to see why it died
if [ -f $logfile ]; then
  log=/tmp/log-weather
  tail -40 $logfile >$log
  /home/jim/scripts/email-log.sh $log "weather log"
  rm $log
  fi

# restart process
pywws-livelog-daemon -v -p $pidfile $datadir $logfile start
```

The process id of the daemon is stored in `pidfile`. If the process is running, the script does nothing. If the process has crashed, it emails the last 40 lines of the log file to me (using a script that creates a message and passes it to sendmail) and then restarts `pywws.livelogdaemon`. You'll need to edit this quite a lot to suit your file locations and so on, but it gives some idea of what to do.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come integrare pywws con vari servizi meteorologici

Questa guida dà brevi istruzioni su come utilizzare pywws con alcuni altri servizi meteorologici e software. Non è completo, e alcuni servizi (come Twitter) sono trattati più dettagliatamente altrove.

YoWindow

YoWindow è un display widget meteo in grado di visualizzare i dati da internet o dalla tua stazione meteo. Per visualizzare i dati della tua stazione pywws devi scrivere in un file locale, in genere ogni 48 secondi quando vengono ricevuti i dati nuovi. Questo è facile da fare:

1. Arrestare tutti i software pywws
2. Copiare il modello di esempio " yowindow.xml " nella directory modelli di testo.
3. Se non hai già fatto, modifica `weather.ini` e imposta la voce `local_files` nella sezione `[paths]` una directory adatta per il vostro file di yowindow.
4. Aggiungere il modello yowindow per le attività di `[live]` in `weather.ini`. Impostare il flag `'L'` così il risultato è copiato in una directory locale invece di essere caricata su un sito ftp:

```
[live]
text = [('yowindow.xml', 'L')]
```

5. Riavviare pywws in registrazione "live".

È possibile controllare se il file è aggiornato ogni 48 secondi usando `more` o `cat` per visualizzare sullo schermo il file.

Infine configurare yowindow per utilizzare questo file. Vedere http://yowindow.com/pws_setup.php per le istruzioni su come effettuare questa operazione.

Twitter

Vedere *Come configurare pywws per pubblicare messaggi su Twitter* per le istruzioni complete.

Other «services»

The remaining weather service uploads are handled by the `pywws.toservice` module. See the module's documentation for general configuration options. The following subsections give further information about some of the available services.

Citizen Weather Observer Program

Nuovo nella versione 14.02.dev1156.

- Web site: http://www.wxqa.com/
- Create account: http://www.wxqa.com/SIGN-UP.html
- API: http://www.wxqa.com/faq.html
- Esempi della sezione `weather.ini`:

```
[cwop]
designator = EW9999
latitude = 5130.06N
longitude = 00008.52E
template = default

[logged]
services = ['cwop', 'underground']

[live]
services = ['cwop', 'underground_rf']
```

or, for radio hams:

```
[cwop]
designator = G4XXX
passcode = xxxxxx
latitude = 5130.06N
longitude = 00008.52E
template = default

[logged]
services = ['cwop_ham', 'underground']

[live]
services = ['cwop_ham', 'underground_rf']
```

Note that the latitude and longitude must be in «LORAN» format and leading zeros are required. See question 3 in the CWOP FAQ for more information.

Licensed radio hams use their callsign as the designator and need a passcode. They should use the service name `cwop_ham` instead of `cwop` when running `pywws.toservice` directly and in the `weather.ini services` entries. (The same `[cwop]` config section is used for both.)

CWOP uploads are rate-limited by pywws, so you can safely add it to both the `[live]` and `[logged]` sections in `weather.ini`.

The CWOP/APRS uploader is based on code by Marco Trevisan <mail@3v1n0.net>.

MQTT

Nuovo nella versione 14.12.0.dev1260.

MQTT is a «message broker» system, typically running on `localhost` or another computer in your home network. Use of MQTT with pywws requires an additional library. See *Dependencies - MQTT* for details.

- MQTT: http://mqtt.org/

- Mosquitto (a lightweight broker): http://mosquitto.org/
- Esempi della sezione `weather.ini`:

```
[mqtt]
topic = /weather/pywws
hostname = localhost
port = 1883
client_id = pywws
retain = True
auth = False
user = unknown
password = unknown
template = default
multi_topic = False

[logged]
services = ['mqtt', 'underground']
```

pywws will publish a JSON string of the data specified in the `mqtt_template_1080.txt` file. This data will be published to the broker running on hostname, with the port number specified. (An IP address can be used instead of a host name.) `client_id` is a note of who published the data to the topic. `topic` can be any string value, this needs to be the topic that a subscriber is aware of.

`retain` is a boolean and should be set to `True` or `False` (or left at the default `unknown`). If set to `True` this will flag the message sent to the broker to be retained. Otherwise the broker discards the message if no client is subscribing to this topic. This allows clients to get an immediate response when they subscribe to a topic, without having to wait until the next message is published.

`auth`, `user` and `password` can be used for MQTT authentication.

`multi_topic` is a boolean and should be set to `True` or `False`. If set to `True` pywws will also publish all the data each as separate subtopics of the configured `topic`; i.e., with the `topic` set to /weather/pywws pywws will also publish the outside temperature to `/weather/pywws/temp_out` and the inside temperature to `/weather/pywws/temp_in`.

If these aren't obvious to you it's worth doing a bit of reading around MQTT. It's a great lightweight messaging system from IBM, recently made more popular when Facebook published information on their use of it.

This has been tested with the Mosquitto Open Source MQTT broker, running on a Raspberry Pi (Raspian OS). TLS (mqtt data encryption) is not yet implemented.

Thanks to Matt Thompson for writing the MQTT code and to Robin Kearney for adding the retain and auth options.

UK Met Office

- Web site: http://wow.metoffice.gov.uk/
- Create account: https://register.metoffice.gov.uk/WaveRegistrationClient/public/newaccount.do?service=weatherobservations
- API: http://wow.metoffice.gov.uk/support/dataformats#automatic
- Esempi della sezione `weather.ini`:

```
[metoffice]
site id = 12345678
aws pin = 987654
template = default
```

```
[logged]
services = ['metoffice', 'underground']
```

Open Weather Map

- Web site: http://openweathermap.org/
- Create account: http://home.openweathermap.org/users/sign_up
- API: http://openweathermap.org/stations#trans
- Esempi della sezione `weather.ini`:

```
[openweathermap]
lat = 51.501
long = -0.142
alt = 10
user = ElizabethWindsor
password = corgi
id = Buck House
template = default

[logged]
services = ['openweathermap', 'underground']
```

When choosing a user name you should avoid spaces (and probably non-ascii characters as well). Having a space in your user name causes strange «internal server error» responses from the server.

The default behaviour is to use your user name to identify the weather station. However, it's possible for a user to have more than one weather station, so there is an optional `name` parameter in the API that can be used to identify the station. This appears as `id` in `weather.ini`. Make sure you choose a name that is not already in use.

PWS Weather

- Web site: http://www.pwsweather.com/
- Creare un account: http://www.pwsweather.com/register.php
- API basate sul protocollo WU: http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol
- Esempi della sezione `weather.ini`:

```
[pwsweather]
station = ABCDEFGH1
password = xxxxxxx
template = default

[logged]
services = ['pwsweather', 'underground']
```

temperatur.nu

- Web site: http://www.temperatur.nu/
- Esempi della sezione `weather.ini`:

```
[temperaturnu]
hash = ???
template = default

[logged]
services = ['temperaturnu', 'underground']
```

You receive the hash value from the temperatur.nu admins during sign up. It looks like «d3b07384d113edec49eaa6238ad5ff00».

Weather Underground

- Creare un account: http://www.wunderground.com/members/signup.asp
- API: http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol
- Esempi della sezione `weather.ini`:

```
[underground]
station = ABCDEFGH1
password = xxxxxxx
template = default

[logged]
services = ['underground', 'metoffice']
```

Weather Underground «RapidFire» updates

Weather Underground ha un secondo upload URL per aggiornamenti in tempo reale appena 2,5 secondi. Se si esegue pywws con "live logging" (vedere *Come impostare una registrazione "live" con pywws*) è possibile utilizzare questa opzione per inviare gli aggiornamenti ogni 48 secondi, con l'aggiunta di "underground_rf" nella sezione `[live]` delle attività `weather.ini`:

```
[underground]
station = ABCDEFGH1
password = xxxxxxx
template = default

[live]
services = ['underground_rf']

[logged]
services = ['underground', 'metoffice']
```

Assicurarsi di che avere attivo un servizio "underground" in `[logged]` o `[hourly]`. In questo modo vengono inviati i record "catchup" per colmare eventuali lacune se vostra stazione passa alla modalità offline per qualche motivo.

wetter.com

- Web site: http://www.wetter.com/wetter_aktuell/wetternetzwerk/
- Register station: http://www.wetter.com/mein_wetter/wetterstation/willkommen/

- Esempi della sezione `weather.ini`:

```
[wetterarchivde]
user_id = 12345
kennwort = ab1d3456i8
template = default

[logged]
services = ['wetterarchivde', 'underground']

[live]
services = ['wetterarchivde', 'underground_rf']
```

Custom Request Headers

The `pywws.toservice` module does support the injection of one or more custom request headers for special cases where you want to integrate with a service that, for example, requires you to pass an authentication key header along with each request, such as `x-api-key`.

These headers can be added to your `a_service.ini` file in the format of key value pairs:

```
[config]
url          = https://my-aws-api-gw.execute-api.eu-west-1.amazonaws.com/test/station
catchup              = 100
interval     = 0
use get              = True
result               = []
auth_type    = None
http_headers         = [('x-api-key', 'my-api-key'), ('x-some-header', 'value')]
```

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come configurare pywws per pubblicare messaggi su Twitter

Installare le dipendenze

Postare su Twitter richiede alcuni software aggiuntivi. Vedere *Dipendenze* - *Postare su Twitter*.

Creare un account su Twitter

Si potrebbero postare aggiornamenti meteo all'account "normale" di Twitter, ma penso che sia meglio avere un conto separato solo per i bollettini meteo. Questo potrebbe essere utile a qualcuno che vive nella tua zona, ma non vuole sapere quello che avevi per la prima colazione.

Autorizzare pywws per inviare al tuo account di Twitter

Se si esegue pywws su un dispositivo di bassa potenza come un router, potrebbe essere più facile per eseguire questo passaggio di autorizzazione su un altro computer, sempre che ci sia ‘‘ python-oauth2 ‘‘ installato. Utilizzare una directory "data" vuota – il file `weather.ini` verrà creato i cui contenuti possono essere copiati nel file `weather.ini` reale utilizzando qualsiasi editor di testo.

Assicurarsi che non vi siano altri software pywws in esecuzione, quindi su esegui *TwitterAuth*:

```
python -m pywws.TwitterAuth ~/weather/data
```

(Replace `~/weather/data` with your data directory.)

In questo modo, si apre una finestra del browser web (o un URL da copiare nel il browser web) dove è possibile accedere al proprio account Twitter e autorizzare pywws. Il browser visualizzerà 7 cifre che è necessario copiare nel programma *TwitterAuth*. Se ha successo, il file `weather.ini` avrà una sezione `[twitter]` con le voci `secret` and `key`. (Non rivelare a nessun altro.)

Aggiungere i dati di localizzazione (opzionale)

Modificare il file `weather.ini` e aggiungere voci `latitude` e `longitude` nella sezione `[twitter]`. Ad esempio:

```
[twitter]
secret = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
key = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
latitude = 51.501
longitude = -0.142
```

Creare un modello

I messaggi Twitter vengono generati utilizzando un modello, proprio come la creazione di file da caricare su un sito Web. Copia il modello di esempio "tweet.txt" directory dei modelli testo, poi testalo:

```
python -m pywws.Template ~/weather/data ~/weather/templates/tweet.txt tweet.txt
cat tweet.txt
```

(Replace `~/weather/data` and `~/weather/templates` with your data and template directories.) If you need to change the template (e.g. to change the units or language used) you can edit it now or later.

Pubblicare il tuo primo Meteo Tweet

Ora tutto è pronto per *ToTwitter* essere eseguito:

```
python -m pywws.ToTwitter ~/weather/data tweet.txt
```

Se questo funziona, il vostro nuovo account Twitter, ha pubblicato il suo primo Meteo report. (È necessario eliminare il file tweet.txt).

Aggiungere aggiornamenti Twitter alla tua attività oraria

Modificare il file `weather.ini` e modificare la sezione `[hourly]`. Ad esempio:

```
[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']
```

Si noti l'uso del flag `'T'` – questo dice a pywws si inviare il risultato a tweet invece di caricarlo sul il sito ftp.

Invece si potrebbe cambiare le sezioni `[logged]`, `[12 hourly]` o `[daily]`, ma credo che ogni `[hourly]` è più appropriato per gli aggiornamenti di Twitter.

Cambiato nella versione 13.06_r1015: Aggiunto il flag `'T'`. Precedentemente i modelli Twitter sono stati indicati separatamente in voci `twitter` in `[hourly]` e di altre sezioni. La sintassi precedente funziona ancora, ma è obsoleta.

Include an image in your tweet

Nuovo nella versione 14.05.dev1216.

You can add up to four images to your tweets by specifying the image file locations in the tweet template. Make the first line of the tweet `media path` where `path` is the absolute location of the file. Repeat for any additional image files. The «tweet_media.txt» example template shows how to do this.

The image could be from a web cam, or for a weather forecast it could be an icon representing the forecast. To add a weather graph you need to make sure the graph is drawn before the tweet is sent. I do this by using two `[cron xxx]` sections in weather.ini:

```
[cron prehourly]
format = 59 * * * *
services = []
plot = [('tweet.png.xml', 'L')]
text = []

[cron hourly]
format = 0 * * * *
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
text = [('tweet_media.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']
```

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come utilizzare pywws in un altra lingua

Introduzione

Some parts of pywws can be configured to use your local language instead of British English. This requires an appropriate language file which contains translations of the various strings used in pywws. The pywws project relies on users to provide these translations.

La documentazione pywws può anche essere tradotta in altre lingue. Questo è ancora molto lavoro, ma potrebbe essere molto utile agli utenti potenziali che non leggono molto bene l'inglese.

Using existing language files

Program strings

There may already be a pywws translation for your preferred language. First you need to choose the appropriate two-letter code from the list at http://www.w3schools.com/tags/ref_language_codes.asp. For example, `fr` is the code for French. Now use the *`pywws.Localisation`* module to do a quick test:

```
python -m pywws.Localisation -t fr
```

Questo dovrebbe produrre output con qualcosa di simile a questo:

```
Locale changed from (None, None) to ('fr_FR', 'UTF-8')
Translation set OK
Locale
  decimal point: 23,2
  date & time: lundi, 17 décembre (17/12/2012 16:00:48)
Translations
  'NNW' => 'NNO'
  'rising very rapidly' => 'en hausse très rapide'
  'Rain at times, very unsettled' => 'Quelques précipitations, très perturbé'
```

This shows that pywws is already able to generate French output, and that your installation is correctly configured. Now you can edit the `language` entry in your *weather.ini* file to use your language code.

If the above test shows no translations into your language then you need to create a new language file, as described below.

Text encodings

The pywws default text encoding is ISO-8859-1, also known as Latin-1. This is suitable for several western European languages but not for some others. If you encounter problems you may need to use a different encoding. See the documentation of `pywws.Template` and `pywws.Plot` for more details.

Documentation

If you have downloaded the pywws source files, or cloned the GitHub repository (see *how to get started with pywws*), you can compile a non-English copy of the documentation. This requires the Sphinx package, see *dependencies*.

First delete the old documentation (if it exists) and then recompile using your language:

```
cd ~/weather/pywws
rm -rf doc
LANG=fr python setup.py build_sphinx
```

Note that the `build_sphinx` command doesn't have a `--locale` (or `-l`) option, so the language is set by a temporary environment variable.

You can view the translated documentation by using a web browser to read the file `~/weather/pywws/doc/html/index.html`.

Writing new language files

There are two ways to write new language files (or update existing ones) – use the Transifex online system or use local files. Transifex is preferred as it allows several people to work on the same language, and makes your work instantly available to others.

To test your translation you will need to have downloaded the pywws source files, or cloned the GitHub repository (see *how to get started with pywws*). You will also need to install the `Babel` package, see *dependencies*.

Using Transifex

If you'd like to use Transifex, please go to the pywws Transifex project, click on «help translate pywws» and create a free account.

Visit the pywws project page on Transifex and click on your language, then click on the «resource» you want to translate. (`pywws` contains the program strings used when running pywws, the others contain strings from the pywws documentation.) This opens a dialog where you can choose to translate the strings online. Please read *Notes for translators* before you start.

When you have finished translating you should use the `transifex-client` program (see *dependencies*) to download files for testing. For example, this command downloads any updated files for the French language:

```
cd ~/weather/pywws
tx pull -l fr
```

Now you are ready to *Test the pywws translations*.

Using local files

If you prefer not to use the Transifex web site you can edit language files on your own computer. This is done in two stages, as follows.

Extract source strings

Program messages are extracted using the `Babel` package:

```
cd ~/weather/pywws
mkdir -p build/gettext
python setup.py extract_messages
```

This creates the file `build/gettext/pywws.pot`. This is a «portable object template» file that contains the English language strings to be translated.

The documentation strings are extracted using the `Sphinx` package:

```
cd ~/weather/pywws
python setup.py extract_messages_doc
```

This creates several `.pot` files in the `build/gettext/` directory.

Create language files

The `sphinx-intl` command is used to convert the `.pot` files to language specific `.po` files:

```
cd ~/weather/pywws
sphinx-intl update --locale-dir src/pywws/lang -p build/gettext -l fr
```

Now you can open the `.po` files in `src/pywws/lang/fr/LC_MESSAGES/` with your favourite text editor and start filling in the empty `msgstr` strings with your translation of the corresponding `msgid` string. Please read *Notes for translators* before you start.

Test the pywws translations

The `Babel` package is used to compile program strings:

```
python setup.py compile_catalog --locale fr
```

(Replace `fr` with the code for the language you are testing.)

After compilation you can test the translation:

```
python setup.py build
sudo python setup.py install
python -m pywws.Localisation -t fr
```

`Sphinx` is used to build the translated documentation:

```
cd ~/weather/pywws
rm -rf doc
LANG=fr python setup.py build_sphinx
```

You can view the translated documentation by using a web browser to read the file `~/weather/pywws/doc/html/index.html`.

Notes for translators

The pywws program strings (`pywws.po`) are quite simple. They comprise simple weather forecasts («Fine weather»), air pressure changes («rising quickly») and the 16 points of the compass («NNE»). Leave the «(%Z)» in «Time (%Z)» unchanged and make sure your translation's punctuation matches the original.

The other files contain strings from the pywws documentation. These are in reStructuredText. This is mostly plain text, but uses characters such as backquotes (‘), colons (:) and asterisks (*) for special purposes. You need to take care to preserve this special punctuation. Do not translate program source, computer instructions and cross-references like these:

```
`pip <http://www.pip-installer.org/>`_
:py:class:`datetime.datetime`
:obj:`ParamStore <pywws.DataStore.ParamStore>`\\ (root_dir, file_name)
pywws.Forecast
``pywws-livelog``
```

Translating all of the pywws documentation is a lot of work. However, when the documentation is «compiled» any untranslated strings revert to their English original. This means that a partial translation could still be useful – I suggest starting with the documentation front page, `index.po`.

Inviare a Jim la traduzione

I'm sure you would like others to benefit from the work you've done in translating pywws. If you've been using Transifex then please send me an email (jim@jim-easterbrook.me.uk) to let me know there's a new translation available. Otherwise, please email me any `.po` files you create.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

weather.ini - configurazione del formato del file

Quasi tutta la configurazione di pywws è tramite un unico file nella directory dei dati: weather.ini. Questo file ha una struttura simile a quella del file Microsoft Windows INI. Esso è diviso in «sections»", ognuna delle quali ha un numero di voci «name = value». L'ordine in cui appaiono spesso delle sezioni non è importante.

Qualsiasi editor di testo può essere utilizzato per modificare il file. (Non provare a modificarlo durante l'esecuzione di qualsiasi altro software pywws). In molti casi sarà pywws inizializzare le voci a valori ragionevoli.

Un altro file, status.ini, viene utilizzato per memorizzare alcune informazioni che pywws utilizza internamente. È descritto alla fine di questo documento. Nell'uso normale non è necessario modificarlo.

Le seguenti sezioni sono attualmente in uso:

- config: varie configurazioni di sistema.
- paths: directory in cui sono memorizzati i modelli ecc.
- live: attività da fare ogni 48 secondi.
- logged: attività da fare ogni volta la stazione registra un record di dati.
- cron: tasks to be done at a particular time or date.
- hourly: attività da fare ogni ora.
- 12 hourly: attività da fare ogni 12 ore.
- daily: attività da fare ogni giorno.
- ftp: configurazione per il caricamento di un sito Web.
- twitter: configurazione per postare su Twitter
- underground, metoffice, temperaturnu etc: configurazione di postare sui "services".

config: varie configurazione di sistema

```
[config]
ws type = 1080
day end hour = 21
pressure offset = 9.4
gnuplot encoding = iso_8859_1
template encoding = iso-8859-1
language = en
logdata sync = 1
rain day threshold = 0.2
asynchronous = False
usb activity margin = 3.0
```

```
gnuplot version = 4.2
frequent writes = False
```

`ws type` è il «tipo» della stazione meteo. Esso deve essere impostato `1080` per la maggior parte delle stazioni meteorologiche, o `3080` se la vostra console stazione visualizza illuminazione solare.

`day end hour` è la fine del «giornata meteorologica», nell'ora locale, senza tenere conto dell'ora legale. Valori tipici sono 21, 9, o 24. È necessario aggiornare tutti i vostri dati memorizzati eseguendo `pywws.Reprocess` per rielaborare tutto dopo aver modificato questo valore.

`pressure offset` è la differenza tra la pressione dell'aria assoluta e relativa (al livello del mare). È copiata dalla stazione meteo, supponendo che è stata impostata fino a visualizzare la corretta pressione relativa, ma è possibile modificare il valore in weather.ini per calibrare la vostra stazione. È necessario aggiornare tutti i vostri dati memorizzati tramite l'esecuzione di `pywws.Reprocess` dopo aver modificato questo valore.

Cambiato nella versione 13.10_r1082: inserito `pressure offset` un elemento di configurazione. In precedenza è sempre stata letta dalla stazione meteo.

`gnuplot encoding` è la codifica del testo utilizzato per tracciare i grafici. Il valore predefinito di `iso_8859_1` che consente di usare il simbolo del grado, il che è utile per un'applicazione meteo! Altri valori potrebbero essere necessari se la lingua comprende i caratteri accentati. I possibili valori dipendono dalla vostra installazione gnuplot così alcuni esperimenti possono essere necessari.

`template encoding` è la codifica del testo utilizzato per i modelli. Il valore predefinito è `iso-8859-1`, che è la codifica utilizzata nell'esempio dei modelli. Se è necessario creare modelli con un diverso set di caratteri, è necessario modificare questo valore in base ai vostri modelli.

`language` è utilizzata per localizzare pywws. E" facoltativa, siccome pywws solitamente utilizza il computer come lingua di default impostata dalla variabile di ambiente LANG. Le lingue disponibili sono quelle della sottodirectory `translations` del vostra installazione pywws. Se si imposta un'altra lingua non presente, pywws tornerà in Inglese.

`logdata sync` imposta la qualità di sincronizzazione utilizzato da *pywws.LogData*. Impostare a 0 è più veloce ma imprecisa o 1 più lento ma preciso.

`rain day threshold` è la quantità di pioggia (in mm) che deve cadere in un giorno per poter qualificare come un giorno di pioggia nei dati di riepilogo mensili. È necessario aggiornare tutti i vostri dati memorizzati tramite l'esecuzione di `pywws.Reprocess` dopo aver modificato questo valore.

Nuovo nella versione 13.09_r1057: `asynchrouous` controlla l'utilizzo di un processo separato in upload di `pywws.LiveLog`.

Nuovo nella versione 13.10_r1094: `usb activity margin` controlla l'algoritmo che evita il problema «USB lockup» che colpisce alcune stazioni. Imposta un numero di secondi su entrambi i lati in attesa di attività della stazione (ricevendo una lettura dall'esterno o la registrazione di una lettura) finchè pywws non ottiene i dati dalla stazione. Se la stazione non è interessato dal problema blocco USB è possibile impostare `usb activity margin` to 0.0.

Nuovo nella versione 13.11_r1102: `gnuplot version` dice `pywws.Plot` e `pywws.WindRose` quale versione di gnuplot è installata sul tuo computer. Questo permette loro di utilizzare le caratteristiche specifiche della versione per dare una migliore qualità di stampa.

Nuovo nella versione 14.01_r1133: `frequent writes` tells `pywws.Tasks` to save weather data and status to file every time there is new logged data. The default is to save the files every hour, to reduce «wear» on solid state memory such as the SD cards used with Raspberry Pi computers. If your weather data directory is stored on a conventional disc drive you can set `frequent writes` to `True`.

paths: directory in cui sono memorizzati modelli ecc.

```
[paths]
templates = /home/$USER/weather/templates/
graph_templates = /home/$USER/weather/graph_templates/
user_calib = /home/$USER/weather/modules/usercalib
work = /tmp/weather
local_files = /home/$USER/weather/results/
```

Queste tre voci specificano dove i tuoi modelli testo e modelli di grafici vengono archiviati, dove devono essere creati i file temporanei, dove l'output dei file (che non sono stati caricati) dovrebbe essere messo e (se ne hai uno) la posizione del tuo modulo di calibrazione.

live: attività da fare ogni 48 secondi

```
[live]
services = ['underground_rf']
text = [('yowindow.xml', 'L')]
plot = []
```

This section specifies tasks that are to be carried out for every data sample during "live logging", i.e. every 48 seconds.

`services` is a list of "services" to upload data to. Each one listed must have a configuration file in `pywws/services/`. See ../api/pywws.toservice for more detail. pywws will automatically limit the frequency of service uploads according to each service's specification.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

Cambiato nella versione 13.05_r1013: aggiunto il modello `'yowindow.xml'`. Precedentemente il file yowindow era generato da un modulo separato, richiamato dalla voce `yowindow` nella sezione `[live]`. La sintassi precedente funziona ancora, ma è obsoleta.

logged: attività da fare ogni volta la stazione registra un record di dati

```
[logged]
services = ['underground', 'metoffice']
text = []
plot = []
```

Questa sezione specifica le attività che devono essere effettuate ogni volta che viene registrato un record di dati in modalità "live logging" o ogni volta che viene eseguito un attività di cron.

`services` è un elenco di "servizi" per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See ../api/pywws.toservice per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

cron: tasks to be done at a particular time or date

Nuovo nella versione 14.05.dev1211.

```
[cron prehourly]
format = 59 * * * *
plot = [('tweet.png.xml', 'L')]
services = []
text = []

[cron hourly]
format = 0 * * * *
plot = ['7days.png.xml', '2014.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt', '2014.txt']
services = []

[cron daily 9]
format = 0 9 * * *
plot = ['28days.png.xml']
text = [('forecast.txt', 'T'), 'forecast_9am.txt', 'forecast_week.txt']
services = []

[cron daily 21]
format = 0 21 * * *
text = ['forecast_9am.txt']
services = []
plot = []

[cron weekly]
format = 0 9 * * 6
plot = ['2008.png.xml', '2009.png.xml', '2010.png.xml', '2011.png.xml',
        '2012.png.xml', '2013.png.xml']
text = ['2008.txt', '2009.txt', '2010.txt', '2011.txt', '2012.txt', '2013.txt']
services = []
```

`[cron name]` sections provide a very flexible way to specify tasks to be done at a particular time and/or date. `name` can be anything you like, but each `[cron name]` section must have a unique name.

To use `[cron name]` sections you need to install the «croniter» package. See *Dipendenze* for more detail.

`format` specifies when the tasks should be done (in local time), in the usual crontab format. (See `man 5 crontab` on any Linux computer.) Processing is not done exactly on the minute, but when the next live or logged data arrives.

hourly: le attività da fare ogni ora

```
[hourly]
services = []
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt', 'feed_hourly.xml']
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
```

Questa sezione specifica le attività che devono essere eseguite ogni ora in modalità “live logging” o l’esecuzione oraria di un attività di cron.

`services` è un elenco di “servizi” per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See ../api/pywws.toservice per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

Cambiato nella versione 13.06_r1015: Aggiunto il flag `'T'`. Precedentemente i modelli Twitter sono stati indicati separatamente in voci `twitter` in `[hourly]` e di altre sezioni. La sintassi precedente funziona ancora, ma è obsoleta.

12 hourly: le attività da fare ogni 12 ore

```
[12 hourly]
services = []
text = []
plot = []
```

This section specifies tasks that are to be carried out every 12 hours when "live logging" or running an hourly cron job. Use it for things that don't change very often, such as monthly graphs. The tasks are done at your day end hour, and 12 hours later.

`services` è un elenco di "servizi" per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See ../api/pywws.toservice per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

daily: le attività da fare ogni 24 ore

```
[daily]
services = []
text = ['feed_daily.xml']
plot = ['2008.png.xml', '2009.png.xml', '2010.png.xml', '28days.png.xml']
```

This section specifies tasks that are to be carried out every day when "live logging" or running an hourly cron job. Use it for things that don't change very often, such as monthly or yearly graphs. The tasks are done at your day end hour.

`services` è un elenco di "servizi" per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See ../api/pywws.toservice per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

ftp: configurazione di caricamento su un sito web

```
[ftp]
local site = False
secure = False
site = ftp.your_isp.co.uk
user = username
password = userpassword
directory = public_html/weather/data/
port = 21
```

Queste voci forniscono i dettagli del sito web (o delle directory locali) in cui processare i file di testo e le immagini grafiche che devono essere trasferite.

`local site` Specifica se il file deve essere copiato in una directory locale o inviato a un sito remoto. È possibile impostare questa opzione se si esegue il server web sulla stessa macchina di pywws.

`secure` specifies whether to transfer files using SFTP (secure FTP) instead of the more common FTP. Your web site provider should be able to tell you if you can use SFTP. Note that you may need to change the `port` value when you change to or from secure mode.

`site` è l'indirizzo web del sito FTP dove trasferire i file.

`user` e `password` sono i dati di login del sito FTP. Il provider del sito web dovrebbe avere comunicato a voi.

`privkey` is the path to a private SSH-key. For SFTP (secure FTP) this can be used for authentication instead of a password, which offers additional benefits in terms of security. When this is used the password-parameter can be left empty.

`directory` specifica il percorso sul sito FTP (o il file system locale) dove devono essere memorizzati i file. Nota che si potrebbe avere la necessità di sperimentare un po" - potrebbe essere necessario un carattere "/" all'inizio del percorso.

Nuovo nella versione 13.12.dev1120: `port` specifies the port number to use. Default value is 21 for FTP, 22 for SFTP. Your web site provider may tell you to use a different port number.

twitter: Configurazione della pubblicazione su Twitter

```
[twitter]
secret = longstringofrandomcharacters
key = evenlongerstringofrandomcharacters
latitude = 51.365
longitude = -0.251
```

`secret` and `key` sono i dati di autenticazione forniti da Twitter. Per averli, usa il programma *`pywws.TwitterAuth`*.

`latitude` and `longitude` sono i dati di posizione facoltativi. Se si include la vostra stazione meteo tweet avrà informazioni sulla posizione in modo che gli utenti possano vedere dov'è la vostra stazione meteo. Potrebbe anche consentire alle persone di trovare la vostra stazione meteo tweet se si cerca per località.

underground, metoffice, temperaturnu ecc: Configurazione della pubblicazione su "servizi"

```
[underground]
station = IXYZABA5
password = secret
```

Queste sezioni contengono informazioni quali password e ID stazione necessari per caricare i dati in servizi meteorologici. I nomi dei dati dipendono dal servizio. L'esempio illustrato è per Weather Underground.

`station` Il PWS ID (identificativo della stazione) assegnato alla stazione meteo da Weather Underground.

`password` è la tua password di Weather Underground.

status.ini - formato del file di stato

Questo file è stato scritto da pywws e non dovrebbe (solitamente) essere modificato. Le seguenti sezioni che sono attualmente in uso:

- fixed: valori copiati dalla stazione meteorologica «fixed block».
- clock: informazioni di sincronizzazione.
- last update: data e ora del completamento delle attività più recenti.

fixed: valori copiati dalla stazione meteorologica «fixed block».

```
[fixed]
fixed block = {...}
```

`fixed block` tutti i dati memorizzati nei primi 256 byte di memoria della stazione. Questo include i valori massimi e minimi, impostazioni di soglia di allarme, unità di visualizzazione e così via.

clock: informazioni di sincronizzazione

```
[clock]
station = 1360322930.02
sensor = 1360322743.69
```

Questi valori sono i tempi misurati quando l'orologio della stazione registra alcuni dati e quando i sensori esterni trasmettono un nuovo set di dati. Essi sono utilizzati per cercare di impedire che l'interfaccia USB si blocca se il computer accede alla stazione meteo al tempo stesso di uno di questi eventi, è un problema comune a molte stazioni compatibile con EasyWeather. I tempi sono misurati ogni 24 ore per consentire la deriva negli orologi.

last update: data e ora del completamento delle attività più recenti

```
[last update]
hourly = 2013-05-30 19:04:15
logged = 2013-05-30 19:04:15
daily = 2013-05-30 09:04:15
openweathermap = 2013-05-30 18:59:15
underground = 2013-05-30 18:58:34
metoffice = 2013-05-30 18:59:15
12 hourly = 2013-05-30 09:04:15
```

Questi record data & ora sono dell'ultimo completamento riuscito delle varie attività. Essi sono utilizzati per consentire alle attività infruttuose (per esempio mancanza di rete prevenzione upload) ad essere riprovata dopo pochi minuti.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Understanding pywws log files

The pywws software uses Python's logging system to report errors, warnings and information that may be useful in diagnosing problems. When you run the software interactively these messages are sent to your terminal window, when running a daemon process or cron job they should be written to a log file.

This document explains some of the pywws logging messages. It's not a complete guide, and the messages you see will depend on your weather station and configuration, but it should help you understand some of the more common ones.

Many pywws commands have a `-v` or `--verbose` option to increase the verbosity of the logging messages. This option can be repeated for even more verbosity, which may be useful when trying to diagnose a particular fault.

Here are some typical logging outputs. The first shows pywws being run interactively:

```
jim@firefly ~ $ pywws-livelog -v ~/weather/data/
08:50:43:pywws.Logger:pywws version 15.07.1.dev1308
08:50:43:pywws.Logger:Python version 2.7.3 (default, Mar 18 2014, 05:13:23) [GCC 4.6.
↪3]
08:50:44:pywws.WeatherStation.CUSBDrive:using pywws.device_libusb1
08:50:49:pywws.Calib:Using user calibration
08:50:49:pywws.Tasks.RegularTasks:Starting asynchronous thread
08:51:05:pywws.ToService(wetterarchivde):1 records sent
08:51:06:pywws.ToService(underground_rf):1 records sent
08:51:06:pywws.ToService(cwop):1 records sent
08:51:52:pywws.ToService(wetterarchivde):1 records sent
08:51:52:pywws.ToService(underground_rf):1 records sent
08:52:40:pywws.ToService(wetterarchivde):1 records sent
08:52:40:pywws.ToService(underground_rf):1 records sent
08:53:28:pywws.ToService(wetterarchivde):1 records sent
08:53:28:pywws.ToService(underground_rf):1 records sent
```

Note that each line begins with a time stamp, in local time. Line 1 is the command used to start pywws. Line 2 shows the pywws version. Line 3 shows the Python version. Line 4 shows which Python USB library is being used. Line 5 shows that a *"user calibration"* routine is being used. Line 6 shows that a separate thread is being started to handle uploads (see *config: varie configurazione di sistema*). The remaining lines show uploads to various weather «services» (see *Come integrare pywws con vari servizi meteorologici*). You can see from the time stamps that they happen at 48 second intervals.

When running pywws as a daemon process the logging is less verbose:

```
2015-07-20 10:46:00:pywws.Logger:pywws version 15.07.1.dev1308
2015-07-20 10:50:06:pywws.weather_station:live_data log extended
2015-07-20 16:25:59:pywws.weather_station:setting station clock 59.637
2015-07-20 16:25:59:pywws.weather_station:station clock drift -0.461377 -0.296364
2015-07-20 16:30:24:pywws.ToService(wetterarchivde):<urlopen error [Errno -2] Name or␣
↪service not known>
2015-07-20 16:30:24:pywws.ToService(underground_rf):<urlopen error [Errno -2] Name or␣
↪service not known>
2015-07-20 23:01:16:pywws.ToService(openweathermap):<urlopen error [Errno -2] Name or␣
↪service not known>
2015-07-21 01:14:18:pywws.weather_station:setting sensor clock 42.6678
2015-07-21 01:14:18:pywws.weather_station:sensor clock drift -2.03116 -1.98475
2015-07-21 09:00:47:pywws.ToTwitter:('Connection aborted.', gaierror(-2, 'Name or␣
↪service not known'))
2015-07-21 09:00:55:pywws.Upload:[Errno -2] Name or service not known
2015-07-21 09:01:05:pywws.ToService(cwop):[Errno -3] Temporary failure in name␣
↪resolution
2015-07-21 09:06:05:pywws.ToService(underground):<urlopen error [Errno -2] Name or␣
↪service not known>
2015-07-21 09:06:05:pywws.ToService(metoffice):<urlopen error [Errno -2] Name or␣
↪service not known>
2015-07-21 16:30:59:pywws.weather_station:setting station clock 59.4771
2015-07-21 16:30:59:pywws.weather_station:station clock drift -0.159373 -0.262116
```

Each line begins with a date and time stamp, in local time. Line 1 shows the pywws version. The remaining lines show normal status messages that are described below.

Clock drift

```
2015-08-31 20:10:45:pywws.weather_station:setting station clock 45.7137
2015-08-31 20:10:45:pywws.weather_station:station clock drift -0.0171086 -0.313699
2015-09-01 01:54:59:pywws.weather_station:setting sensor clock 35.2755
2015-09-01 01:54:59:pywws.weather_station:sensor clock drift -1.12118 -1.37694
```

These lines report how the weather station's internal («station») and external («sensor») clocks are drifting with respect to the computer's clock. These measurements are used to avoid accessing the station's USB port at the same time as it is receiving data or logging data, as this is known to cause some station's USB ports to become inaccessible. The two «drift» figures are the current value (only accurate to about 1 second) and the long term average. You should ensure that the `usb activity margin` value in your *weather.ini file* is at least 0.5 seconds greater than the absolute value of the long term drift of each clock. Note that these drift values change with temperature.

The clock drifts are measured at approximately 24 hour intervals. If pywws loses synchronisation with your station it will measure them again. Doing this measurement increases the risk of causing a USB lockup, so if pywws often loses synchronisation you should try and find out why it's happening.

Network problems

Occasionally one or more of the services and web sites you upload data to may become unavailable. This leads to error messages like these:

```
2015-08-03 04:19:49:pywws.ToService(underground_rf):[Errno 104] Connection reset by␣
↪peer
2015-08-03 04:49:27:pywws.ToService(underground_rf):<urlopen error [Errno -2] Name or␣
↪service not known>
2015-08-03 05:19:41:pywws.ToService(wetterarchivde):<urlopen error [Errno 101]␣
↪Network is unreachable>
2015-08-03 05:19:46:pywws.ToService(underground_rf):<urlopen error [Errno 101]␣
↪Network is unreachable>
2015-08-03 05:50:52:pywws.ToService(wetterarchivde):<urlopen error [Errno -2] Name or␣
↪service not known>
2015-08-03 05:50:52:pywws.ToService(underground_rf):<urlopen error [Errno -2] Name or␣
↪service not known>
```

To avoid swamping the log files duplicate messages are not logged, so you cannot tell how long the network outage lasted from the log files.

Status

```
2015-09-01 21:50:21:pywws.weather_station:status {'unknown': 0, 'invalid_wind_dir':␣
↪2048, 'lost_connection': 64, 'rain_overflow': 0}
```

The raw weather station data includes some «status» bits. If any of these bits is non-zero when pywws starts, or the status changes value when pywws is running, the status value is logged. The most common problem is `lost_connection`: the weather station console is not receiving data from the outside sensors. Contact is often restored a few minutes later, but if not you may need to reset your weather station console by taking its batteries out. The `invalid_wind_dir` bit indicates that the wind direction sensor value is missing or invalid. The `rain_overflow` bit is set when the rain gauge counter has reached its maximum value and gone back to zero.

Please let me know if you ever get a non-zero value for `unknown`, particularly if you are able to correlate it with some other event. There are 6 bits of data in the status byte whose function is not yet known.

Log extended

```
2015-08-10 08:25:59:pywws.weather_station:live_data log extended
2015-08-10 08:41:59:pywws.weather_station:live_data log extended
2015-08-10 08:57:59:pywws.weather_station:live_data log extended
```

This shows a curiosity in the weather station's internal processing. As the internal and external sensors drift there comes a time when an external reading is expected at the same time as the station is due to log some data. To avoid a clash the station delays logging by one minute. As the external readings are at 48 second intervals this avoids the problem until 16 minutes later (with the normal 5 minute logging interval) when another one minute delay is needed. Eventually the clocks drift apart and normal operation is resumed.

Rain reset

```
2015-08-25 13:30:51:pywws.Process.HourAcc:2015-08-25 12:30:48 rain reset 1048.4 ->␣
↪1047.1
2015-08-25 13:35:51:pywws.Process.HourAcc:2015-08-25 12:30:48 rain reset 1048.4 ->␣
↪1047.1
2015-08-25 13:40:51:pywws.Process.HourAcc:2015-08-25 12:30:48 rain reset 1048.4 ->␣
↪1047.1
```

The raw rainfall data from the outside sensors is the total number of times the «see saw» has tipped since the external sensors were last reset (by a battery change, unless you do it quickly). This number should only ever increase, so the `pywws.Process` module warns of any decrease in the value as it may indicate corrupted data that needs manually correcting. The logging message includes the UTC time stamp of the problem data to help you find it.

Live data missed

```
1 2015-10-30 04:48:19:pywws.ToService(underground_rf):1 records sent
2 2015-10-30 04:49:07:pywws.ToService(underground_rf):1 records sent
3 2015-10-30 04:49:56:pywws.weather_station:live_data missed
4 2015-10-30 04:50:44:pywws.ToService(underground_rf):1 records sent
5 2015-10-30 04:51:31:pywws.ToService(underground_rf):1 records sent
```

Line 3 indicate that pywws failed to capture live data.

There are two possible causes. One is that a new data record is identical to the previous one so pywws doesn't detect a change. This is unlikely to happen if you are receiving wind data properly.

The more likely reason is that processing the previous record took so long that the next one arrived when pywws wasn't ready for it. «Processing» can include uploading to the Internet which is often prone to delays. A solution to this is to set «asynchronous» to True in weather.ini. This uses a separate thread to do the uploading.

You may run with higher verbosity to get more information. The «pause» values should indicate how soon it's ready for the next data.

Note that this is just an occasional missing «live» record though, so if it does not happen often you shouldn't worry too much about it.

«Live log» synchronisation

If you run pywws at a high verbosity you may see messages like the following:

```
10:26:05:pywws.Logger:pywws version 15.07.0.dev1307
10:26:05:pywws.Logger:Python version 2.7.8 (default, Sep 30 2014, 15:34:38) [GCC]
10:26:05:pywws.WeatherStation.CUSBDrive:using pywws.device_libusb1
10:26:06:pywws.Calib:Using user calibration
10:26:06:pywws.Tasks.RegularTasks:Starting asynchronous thread
10:26:06:pywws.weather_station:read period 5
10:26:06:pywws.weather_station:delay 2, pause 0.5
10:26:07:pywws.weather_station:delay 2, pause 0.5
10:26:08:pywws.weather_station:delay 2, pause 0.5
10:26:08:pywws.weather_station:delay 2, pause 0.5
10:26:09:pywws.weather_station:delay 2, pause 0.5
10:26:10:pywws.weather_station:delay 2, pause 0.5
10:26:10:pywws.weather_station:delay 2, pause 0.5
10:26:11:pywws.weather_station:delay 2, pause 0.5
10:26:12:pywws.weather_station:delay 2, pause 0.5
10:26:12:pywws.weather_station:delay 2, pause 0.5
10:26:13:pywws.weather_station:delay 2, pause 0.5
10:26:14:pywws.weather_station:delay 2, pause 0.5
10:26:14:pywws.weather_station:live_data new data
10:26:14:pywws.weather_station:setting sensor clock 38.7398
10:26:14:pywws.weather_station:delay 3, pause 45.4993
10:26:16:pywws.Tasks.RegularTasks:Doing asynchronous tasks
10:27:00:pywws.weather_station:delay 3, pause 0.5
10:27:00:pywws.weather_station:avoid 3.83538614245
10:27:04:pywws.weather_station:live_data new data
10:27:04:pywws.weather_station:delay 3, pause 43.3316
10:27:06:pywws.Tasks.RegularTasks:Doing asynchronous tasks
10:27:48:pywws.weather_station:delay 3, pause 0.5
10:27:48:pywws.weather_station:avoid 3.79589626256
10:27:52:pywws.weather_station:live_data new data
10:27:52:pywws.weather_station:delay 4, pause 0.5
10:27:53:pywws.weather_station:delay 4, pause 0.5
10:27:54:pywws.weather_station:delay 4, pause 0.5
10:27:54:pywws.weather_station:delay 4, pause 0.5
10:27:54:pywws.Tasks.RegularTasks:Doing asynchronous tasks
10:27:55:pywws.weather_station:delay 4, pause 0.5
10:27:56:pywws.weather_station:delay 4, pause 0.5
10:27:56:pywws.weather_station:delay 4, pause 0.5
10:27:57:pywws.weather_station:delay 4, pause 0.5
10:27:58:pywws.weather_station:delay 4, pause 0.5
10:27:58:pywws.weather_station:delay 4, pause 0.5
10:27:59:pywws.weather_station:delay 4, pause 0.5
10:28:00:pywws.weather_station:delay 4, pause 0.5
10:28:00:pywws.weather_station:delay 4, pause 0.5
10:28:01:pywws.weather_station:delay 4, pause 0.5
10:28:02:pywws.weather_station:delay 4, pause 0.5
10:28:02:pywws.weather_station:delay 4, pause 0.5
10:28:03:pywws.weather_station:delay 4, pause 0.5
10:28:04:pywws.weather_station:delay 4, pause 0.5
10:28:04:pywws.weather_station:delay 4, pause 0.5
10:28:05:pywws.weather_station:delay 4, pause 0.5
10:28:06:pywws.weather_station:delay 4, pause 0.5
10:28:06:pywws.weather_station:delay 4, pause 0.5
10:28:07:pywws.weather_station:live_data new ptr: 007320
10:28:07:pywws.weather_station:setting station clock 7.43395
10:28:07:pywws.weather_station:avoid 1.91954708099
10:28:10:pywws.DataLogger:1 catchup records
10:28:10:pywws.Process:Generating summary data
```

```
10:28:10:pywws.Process:daily: 2015-08-31 21:00:00
10:28:10:pywws.Process:monthly: 2015-07-31 21:00:00
10:28:10:pywws.Process:monthly: 2015-08-31 21:00:00
10:28:10:pywws.weather_station:delay 0, pause 26.121
10:28:12:pywws.Tasks.RegularTasks:Doing asynchronous tasks
```

Line 6 shows that the weather station has the usual 5 minute logging interval. Lines 7 to 18 show pywws waiting for the station to receive data from the outside sensors. The `delay` value is the number of minutes since the station last logged some data. The `pause` value is how many seconds pywws will wait before fetching data from the station again. Lines 19 & 20 show new data being received and the «sensor» clock being set. Line 21 shows that pywws now knows when data is next expected, so it can sleep for 43 seconds. Line 22 shows the separate «upload» thread doing its processing while the main thread is sleeping. Line 24 shows pywws avoiding USB activity around the time the station should receive external data. Lines 31 to 53 show pywws waiting for the station to log data. Lines 54 & 55 show the station logging some data and pywws using this to set the «station» clock. The 6 digit number at the end of line 54 is the hexadecimal address where «live» data will now be written to, leaving data at the previous address as a «logged» value. Lines 57 to 61 show pywws fetching logged data from the station and then processing it to produce the various summaries.

Crash with traceback

Sometimes pywws software crashes. When it does, the log file will often contain a traceback like this:

```
18:50:57:pywws.LiveLog:error sending control message: Device or resource busy
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/pywws/LiveLog.py", line 80, in LiveLog
    logged_only=(not tasks.has_live_tasks())):
  File "/usr/local/lib/python2.7/dist-packages/pywws/LogData.py", line 256, in live_
↪data
    for data, ptr, logged in self.ws.live_data(logged_only=logged_only):
  File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 446, in␣
↪live_data
    new_ptr = self.current_pos()
  File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 585, in␣
↪current_pos
    self._read_fixed_block(0x0020), self.lo_fix_format['current_pos'])
  File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 641, in␣
↪_read_fixed_block
    result += self._read_block(mempos)
  File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 629, in␣
↪_read_block
    new_block = self.cusb.read_block(ptr)
  File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 265, in␣
↪read_block
    if not self.dev.write_data(buf):
  File "/usr/local/lib/python2.7/dist-packages/pywws/device_pyusb.py", line 152, in␣
↪write_data
    usb.REQ_SET_CONFIGURATION, buf, value=0x200, timeout=50)
USBError: error sending control message: Device or resource busy
```

Line 1 shows the exception that caused the crash. Lines 3 to 18 show where in the program the problem happened. Usually the last one is of interest, but the other function calls show how we got there. Line 19 shows the full exception. In this case it's a USBError raised by the pyusb library.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Indice di umidità (Humidex)

Autore della sezione: Rodney Persky

Premessa

Utilizzare la tua stazione meteo può essere divertente e la segnalazione giornaliera nei siti dei vari dati meteo possono essere molto utili per i vostri vicini per controllare il tempo. Tuttavia, ad un certo punto si potrebbe voler sapere quali effetti ha tempo sul tuo corpo, e se c'è un modo di dire quando è bene o no lavorare all'aperto.

Qui si inserisce in un mondo tutto di calcoli basati sul trasferimento di energia attraverso pareti, e la resistenza da loro offerta. Essa può essere una grande avventura della conoscenza, e consente di risparmiare molto denaro, energia che si muove intorno.

Introduzione

Humidex è uno strumento per determinare in che modo il corpo di un individuo reagirà alla combinazione di vento, umidità e temperatura. Sullo sfondo del quale c'è l'equilibrio termico tra la vita e la pelle, e" gratuito per ISO 7243 «Ambienti caldi - Valutazione dello stress termico sul lavoro l'uomo». Alcune note importanti,

- Questi indici sono basati su un certo numero di ipotesi che possono risultare in sopra o sottovalutazione del tuo stato interno del corpo
- Una stazione meteo personale potrebbe non mostrare le corrette condizioni, e possono avere una sovra o sottostima dell'umidità, vento, temperatura
- Scelte di abbigliamento personale, effetto della stanchezza e la capacità del fisico di respingere il calore, un basso indice di umidità non significa che si può indossare qualsiasi cosa
- Un individuale idoneità effettuerà la propria risposta del fisico alla temperatura che cambia, e l'esperienza sarà di aiuto nel sapere quando smettere di lavorare
- La durata delle attività che possono essere eseguite richiede conoscenze sull'intensità, che non può essere rappresentata da questo indice

Ipotesi

Ci sono un certo numero di ipotesi che sono state fatte per fare questo lavoro che influenzerà direttamente la sua utilizzabilità. Queste ipotesi tuttavia non sono state messe a disposizione da Environment Canada, che sono gli sviluppatori originali dell'Humidex utilizzato nel PYWWS funzione cadhumidex. Tuttavia è abbastanza sicuro nel dire che sarebbe sono state eseguite alcune ipotesi:

- Tipo di abbigliamento, spessore
- Zona di pelle esposta all'aria libera
- Esposizione al sole

Tuttavia, ci sono un certo numero di ipotesi che pywws deve fare nel calcolo Humidex:

- Le letture di temperatura, vento e umidità sono corrette

Ci sono anche ipotesi circa il tipo di fisico individuale e di "acclimatazione"

- Un individuale idoneità effettuerà la risposta del fisico alla temperatura che cambia
- L'esperienza sarà di aiuto nel sapere quando smettere di lavorare

Importanti riferimenti

Corso di preparazione per l'estate - http://www.ec.gc.ca/meteo-weather/default.asp?lang=En&n=86C0425B-1

Come utilizzare

La funzione descrittivamente è denominata `cadhumidex` e ha i parametri di temperatura e di umidità, essenzialmente la funzione opera come una conversione e può essere utilizzata in maniera diretta

```
<ycalc>cadhumidex(data['temp_out'],data['hum_out'])</ycalc>
```

Mettendo insieme, ho aggiunto i colori che seguono i colori base di avvertimento e le diverse staffe per produrre un grafico decente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph>
  <title>Humidity Index, Bands indicate apparent discomfort in standard on-site␣
↪working conditions</title>
  <size>1820, 1024</size>
  <duration>hours=48</duration>
  <xtics>2</xtics>
  <xformat>%H%M</xformat>
  <dateformat></dateformat>
  <plot>
    <yrange>29, 55</yrange>
    <y2range>29, 55</y2range>
    <ylabel></ylabel>
    <y2label>Humidex</y2label>
    <source>raw</source>
    <subplot>
      <title>Humidex</title>
      <ycalc>cadhumidex(data['temp_out'],data['hum_out'])</ycalc>
      <colour>4</colour>
      <axes>x1y2</axes>
    </subplot>
    <subplot>
      <title>HI > 54, Heat Stroke Probable</title>
      <ycalc>54</ycalc>
      <axes>x1y2</axes>
      <colour>1</colour>
    </subplot>
    <subplot>
      <title>HI > 45, Dangerous</title>
      <ycalc>45</ycalc>
      <axes>x1y2</axes>
      <colour>8</colour>
    </subplot>
    <subplot>
      <title>HI > 40, Intense</title>
      <ycalc>40</ycalc>
      <axes>x1y2</axes>
      <colour>6</colour>
    </subplot>
    <subplot>
      <title>HI > 35, Evident</title>
      <ycalc>35</ycalc>
```

```
      <axes>x1y2</axes>
      <colour>2</colour>
    </subplot>
    <subplot>
      <title>HI > 30, Noticeable</title>
      <ycalc>30</ycalc>
      <axes>x1y2</axes>
      <colour>3</colour>
    </subplot>
  </plot>
</graph>
```

Non hai eseguito l'aggiornamento più recente?

Se non si esegue l'aggiornamento più recente o non vuoi, allora questo può essere implementato utilizzando una <ycalc> come segue:

```
<ycalc>data['temp_out']+0.555*(6.112*10**(7.5*data['temp_out']/(237.7+data['temp_out
↪']))*data['hum_out']/100-10)</ycalc>
```

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

3.1.6 Programmi Python e moduli

Set up and configure pywws

pywws.TestWeatherStation	Testare la connessione alla stazione meteo.
pywws.SetWeatherStation	Set some weather station parameters.
pywws.version	Display pywws version information.
pywws.Reprocess	
pywws.TwitterAuth	Autorizzare pywws per inviare al vostro account Twitter
pywws.USBQualityTest	Verificare la qualità della connessione USB della stazione meteo
pywws.EWtoPy	

pywws.TestWeatherStation

Testare la connessione alla stazione meteo.

This script can also be run with the `pywws-testweatherstation` command.

```
usage: python -m pywws.TestWeatherStation [options]
options are:
        --help       display this help
 -c   | --change     display any changes in "fixed block" data
```

```
 -d   | --decode     display meaningful values instead of raw data
 -h n | --history n  display the last "n" readings
 -l   | --live       display 'live' data
 -m   | --logged     display 'logged' data
 -u   | --unknown    display unknown fixed block values
 -v   | --verbose    increase amount of reassuring messages
                     (repeat for even more messages e.g. -vvv)
```

This is a simple utility to test communication with the weather station. If this doesn't work, then there's a problem that needs to be sorted out before trying any of the other programs. Likely problems include not properly installing the USB libraries, or a permissions problem. The most unlikely problem is that you forgot to connect the weather station to your computer!

Funzioni

main([argv])	
raw_dump(pos, data)	

pywws.TestWeatherStation.**raw_dump**(*pos*, *data*)

pywws.TestWeatherStation.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.SetWeatherStation

Set some weather station parameters.

This script can also be run with the `pywws-setweatherstation` command.

```
usage: python -m pywws.SetWeatherStation [options]
options are:
 -h   | --help           display this help
 -c   | --clock          set weather station clock to computer time
                         (unlikely to work)
 -p f | --pressure f     set relative pressure to f hPa
 -r n | --read_period n  set logging interval to n minutes
 -v   | --verbose        increase error message verbosity
 -z   | --zero_memory    clear the weather station logged reading count
```

Funzioni

bcd_encode(value)	
main([argv])	

pywws.SetWeatherStation.**bcd_encode**(*value*)

pywws.SetWeatherStation.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.version

Display pywws version information.

This script can also be run with the `pywws-version` command.

```
usage: python -m pywws.version [options]
options are:
 -h      or --help      display this help
 -v      or --verbose   show verbose version information
```

Funzioni

main([argv])	

pywws.version.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.TwitterAuth

Autorizzare pywws per inviare al vostro account Twitter

```
usage: python -m pywws.TwitterAuth [options] data_dir
options are:
 -h or --help       display this help
data_dir is the root directory of the weather data
```

Questo programma autorizza *pywws.ToTwitter* per postare su un account Twitter. È necessario creare un account prima di eseguire `TwitterAuth`. Si apre una finestra del browser web (o un URL da copiare nel il browser web) che consente di accedere al tuo account Twitter. Se il login ha successo il browser visualizzerà un numero di 7 cifre, numero che poi copia in `TwitterAuth`.

Vedere *Come configurare pywws per pubblicare messaggi su Twitter* per ulteriori dettagli su come usare Twitter con pywws.

Funzioni

TwitterAuth(params)	
main([argv])	

pywws.TwitterAuth.**TwitterAuth**(*params*)

pywws.TwitterAuth.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.USBQualityTest

Verificare la qualità della connessione USB della stazione meteo

```
usage: python -m pywws.USBQualityTest [options]
options are:
 -h | --help           display this help
 -v | --verbose        increase amount of reassuring messages
                       (repeat for even more messages e.g. -vvv)
```

Il collegamento USB per la mia stazione meteo non è affidabile al 100%. La lettura dei dati della stazione dal computer è occasionalmente corrotto, forse per interferenza. Ho cercato di risolvere questo problema mettendo ferrite intorno al cavo USB e trasferire possibili sorgenti di interferenza, come dischi rigidi esterni. Tutti senza successo finora.

Questo programma mette alla prova la connessione USB per gli errori per leggere in modo continuo tutta la memoria della stazione meteo (ad eccezione di quelle parti che possono variare) alla ricerca di errori. Ogni blocco di 32-byte viene letto due volte, e se i due valori differiscono viene visualizzato un messaggio di avviso. Sono visualizzati anche il numero di blocchi letti, e il numero di errori riscontrati.

I typically get one or two errors per hour, so the test needs to be run for several hours to produce a useful measurement. Note that other software that accesses the weather station (such as `pywws.Hourly` or `pywws.LiveLog`) must not be run while the test is in progress.

Se si esegue questo test e non trovate gli errori, per favore fatemelo sapere. C'è qualcosa di miracoloso nella configurazione e mi piacerebbe sapere cosa è!

Funzioni

main([argv])	

pywws.USBQualityTest.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

Get data and process it

pywws.Hourly	
pywws.LiveLog	
pywws.livelogdaemon	

«Internal» modules

pywws.Tasks	
pywws.LogData	Salva i dati della stazione meteo nel file

Continued on next page

Tabella 3.8 – continued from previous page

pywws.Process	
pywws.calib	Calibra i dati grezzi stazione meteo
pywws.Plot	
pywws.WindRose	
pywws.Template	
pywws.Forecast	
pywws.ZambrettiCore	
pywws.Upload	Carica i file su un server web tramite ftp o copiarli in una directory locale
pywws.ToTwitter	Postare messaggi su Twitter
pywws.toservice	
pywws.YoWindow	
pywws.WeatherStation	Ottenere i dati da stazioni meteo compatibili con WH1080/WH3080.
pywws.device_libusb1	Low level USB interface to weather station, using python-libusb1.
pywws.device_pyusb1	Low level USB interface to weather station, using PyUSB v1.0.
pywws.device_pyusb	Low level USB interface to weather station, using PyUSB v0.4.
pywws.device_ctypes_hidapi	Interfaccia di basso livello USB della stazione meteo, utilizzando ctypes per accedere a hidapi.
pywws.device_cython_hidapi	Interfaccia di basso livello USB della stazione meteo, tramite cython-hidapi.
pywws.DataStore	DataStore.py - memorizza le letture in file di facile accesso
pywws.TimeZone	
pywws.Localisation	Localisation.py - fornisce traduzioni di stringhe in lingua locale
pywws.conversions	
pywws.Logger	Codice comune per la registrazione di informazioni ed errori.
pywws.constants	Bits of data used in several places.

pywws.LogData

Salva i dati della stazione meteo nel file

```
usage: python -m pywws.LogData [options] data_dir
options are:
 -h   | --help     display this help
 -c   | --clear    clear weather station's memory full indicator
 -s n | --sync n   set quality of synchronisation to weather station (0 or 1)
 -v   | --verbose  increase number of informative messages
data_dir is the root directory of the weather data
```

This module gets data from the weather station's memory and stores it to file. Each time it is run it fetches all data that is newer than the last stored data, so it only needs to be run every hour or so. As the weather station typically stores two weeks" readings (depending on the logging interval), *pywws.LogData* could be run quite infrequently if you don't need up-to-date data.

There is no date or time information in the raw weather station data, so *pywws.LogData* creates a time stamp for each reading. It uses the computer's clock, rather than the weather station clock which can not be read accurately by the computer. A networked computer should have its clock set accurately by ntp.

La sincronizzazione con la stazione meteo è realizzata in attesa di un cambiamento nei dati attuali. Ci sono due livelli di sincronizzazione, impostate nel file `weather.ini` la sezione `[config]` l'opzione `logdata sync =`. Il livello 0 è più veloce, ma è poco preciso scarta circa dodici secondi. Livello 1 attende dalla stazione meteo che memorizza un nuovo record di dati e ottiene il timestamp accurato di un paio di secondi. Si noti che questo potrebbe richiedere molto tempo, se l'intervallo di registrazione supera i cinque minuti consigliati.

Dettagli API

Funzioni

main([argv])	

Classi

DataLogger(params, status, raw_data)	

class pywws.LogData.**DataLogger**(*params*, *status*, *raw_data*)

check_fixed_block()

catchup(*last_date*, *last_ptr*)

log_data(*sync=None*, *clear=False*)

live_data(*logged_only=False*)

pywws.LogData.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.calib

Calibra i dati grezzi stazione meteo

Questo modulo permette l'adattamento dei dati grezzi(raw) della stazione meteo come parte del passo "processing" (vedi: doc:*pywws.Process*). Ad esempio, se abbiamo montato un imbuto a doppia zona di raccolta del pluviometro, è possibile scrivere una routine di calibrazione per raddoppiare il valore di pioggia.

The default calibration generates the relative atmospheric pressure. Any user calibration you write must also do this.

Scrivendo il vostro modulo di calibrazione

In primo luogo, decidere dove si desidera salvare il vostro modulo. Come i modelli di testo e grafici, è meglio tenerlo separato dal codice pywws, quindi esso non è influenzato da aggiornamenti pywws. Vi suggerisco di creare una directory `modules` nello stesso luogo come la directory `templates`.

Si potrebbe iniziare copiando uno dei moduli calibrazione di esempio, oppure è possibile creare un file di testo nella directory `modules`, ad esempio `calib.py` e copiare il seguente testo in esso:

```
class Calib(object):
    def __init__(self, params, stored_data):
        self.pressure_offset = eval(params.get('config', 'pressure offset'))

    def calib(self, raw):
        result = dict(raw)
        # calculate relative pressure
        result['rel_pressure'] = result['abs_pressure'] + self.pressure_offset
        return result
```

La classe *Calib* ha due metodi. `Calib.__init__()` è il costruttore ed è un buon posto per impostare tutte le costanti necessarie. E" un riferimento passato per la memorizzazione dei dati grezzi, che può essere utile per attività avanzate come la rimozione picchi. `Calib.calib()` genera un singolo set di dati "calibrato" da un singolo set di dati "raw". Ci sono alcune regole da seguire quando si scrive questo file:

- Assicurati di includere la riga `result = dict(raw)`, che copia tutti i dati grezzi(raw) per il risultato valido, dall'inizio.
- Non modificare i dati grezzi(raw).
- Assicurarsi di impostare `result['rel_pressure']`.
- Non dimenticate `return` situato alla fine del file.

Quando hai finito di scrivere il tuo modulo di calibrazione è possibile ottenere da pywws che la utilizzi mettere la sua posizione nel vostro file `weather.ini` Va in sezione `[paths]`, come mostrato nell'esempio di seguito:

```
[paths]
work = /tmp/weather
templates = /home/jim/weather/templates/
graph_templates = /home/jim/weather/graph_templates/
user_calib = /home/jim/weather/modules/usercalib
```

Si noti che il valore `user_calib` non comprende l'estenzione `.py` alla fine del nome del file.

Classi

Calib(params, stored_data)	Classe di taratura che implementa il default o la calibrazione da parte dell'utente.
DefaultCalib(params, stored_data)	Classe di calibrazione predefinita.

class pywws.calib.**DefaultCalib**(*params*, *stored_data*)

Classe di calibrazione predefinita.

This class sets the relative pressure, using a pressure offset originally read from the weather station. This is the bare minimum "calibration" required.

calib(*raw*)

class pywws.calib.**Calib**(*params*, *stored_data*)

Classe di taratura che implementa il default o la calibrazione da parte dell'utente.

Altri moduli pywws utilizzano questo metodo per creare un oggetto di taratura. Il costruttore crea un oggetto predefinito calibrazione o un oggetto di calibrazione dell'utente, secondo il valore di `user_calib` nella sezione `[paths]` dei parametri `params`. Allora adotta la calibrazione oggetto `calib()` metodo come propri.

calibrator = None

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.ZambrettiCore

Funzioni

ZambrettiCode(pressure, month, wind, trend)	Semplice implementazione dell'algoritmo previsioni del tempo di Zambretti.
ZambrettiText(letter)	
main([argv])	

pywws.ZambrettiCore.**ZambrettiCode**(*pressure*, *month*, *wind*, *trend*, *north=True*, *baro_top=1050.0*, *baro_bottom=950.0*)

Semplice implementazione dell'algoritmo previsioni del tempo di Zambretti. Ispirato dall'algoritmo Java di beteljuice.com, e convertito in Python da honeysucklecottage.me.uk, ulteriori informazioni da http://www.meteormetrics.com/zambretti.htm

pywws.ZambrettiCore.**ZambrettiText**(*letter*)

pywws.ZambrettiCore.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.Upload

Carica i file su un server web tramite ftp o li copia in una directory locale

```
usage: python -m pywws.Upload [options] data_dir file [file...]
options are:
 -h or --help    display this help
data_dir is the root directory of the weather data
file is a file to be uploaded
```

I dettagli login e ftp vengono letti dal file weather.ini in data_dir.

Introduzione

Questo modulo che carica il file (in genere) di un sito Web *tramite* ftp/sftp o copia i file in una directory locale (ad esempio se si esegue pywws sul proprio server web). Dettagli della destinazione upload sono memorizzati nel file `weather.ini` nella directory dati. L'unico modo per impostare questi dettagli è modificare il file. Eseguire *pywws.Upload* una volta per impostare i valori predefiniti, che poi possono cambiare. Ecco cosa è probabile trovare quando si modifica la `weather.ini`:

```
[ftp]
secure = False
directory = public_html/weather/data/
```

```
local site = False
password = secret
site = ftp.username.your_isp.co.uk
user = username
```

Queste sono, spero, abbastanza ovvie. Il `local site` consente di passare dal caricamento in un sito remoto alla copia di un sito locale. Se si imposta `local site = True` quindi è possibile eliminare le linee `secure`, `site`, `user` e `password`.

`directory``è il nome di una directory in cui verranno messi tutti i file caricati. Ciò dipenderà dalla struttura del sito web e l'ordinamento dell'host che si utilizza. Il provider di hosting dovrebbe essere in grado di dirvi quali dettagli ``site` and `user` utilizzare. Si dovrebbe già aver scelto una `password`.

L'opzione `secure` consente di passare da ftp normale a sftp (ftp sopra ssh). Alcuni fornitori di hosting offrono questo come un meccanismo di caricamento più sicuro, quindi si dovrebbe probabilmente usare se disponibile.

Dettagli API

Funzioni

main([argv])

Classi

Upload(params)

class pywws.Upload.**Upload**(*params*)

connect()

upload_file(*file*)

disconnect()

upload(*files*)

pywws.Upload.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.ToTwitter

Postare messaggi su Twitter

```
usage: python -m pywws.ToTwitter [options] data_dir file
options are:
 -h | --help  display this help
data_dir is the root directory of the weather data
file is the text file to be uploaded
```

Questo modulo invia un breve messaggio a Twitter. Prima di pubblicare su Twitter è necessario configurare un account e quindi autorizzare pywws eseguendo il programma `TwitterAuth`. Vedere *Come configurare pywws per pubblicare messaggi su Twitter* per le istruzioni dettagliate.

Funzioni

main([argv])

Classi

PythonTwitterHandler(key, secret, latitude, ...)
ToTwitter(params)
TweepyHandler(key, secret, latitude, longitude)

class pywws.ToTwitter.**TweepyHandler**(*key*, *secret*, *latitude*, *longitude*)

post(*status*, *media*)

class pywws.ToTwitter.**PythonTwitterHandler**(*key*, *secret*, *latitude*, *longitude*, *timeout*)

post(*status*, *media*)

class pywws.ToTwitter.**ToTwitter**(*params*)

Upload(*tweet*)

UploadFile(*file*)

pywws.ToTwitter.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.WeatherStation

Ottenere i dati da stazioni meteo compatibili con WH1080/WH3080.

Derivato da wwsr.c by Michael Pendec (michael.pendec@gmail.com), wwsrdump.c by Svend Skafte (svend@skafte.net), modificato da Dave Wells, e altre fonti.

Introduzione

Questo è il modulo che in realtà comunica con l'unità di base della stazione meteo. Non ho molta comprensione dell" USB, quindi copiato molto dal programma in C wwsr di Michael Pendec's

La memoria della stazione meteo è divisa in due parti: «fixed block» di 256 byte e un buffer circolare di 65280 byte. Siccome ogni stringa prende 16 byte la stazione può immagazzinare 4080 letture, o 14 giorni di letture di intervallo di 5 minuti. (Le stazioni tipo 3080 memorizzano 20 byte per la stringa, quindi memorizzano un massimo di 3264). Siccome lettura dei dati è in blocchi di 32 byte, ma ogni lettura meteo è 16 o 20 byte, una piccola cache

viene utilizzata per ridurre il traffico USB. Il comportamento di memorizzazione nella cache può essere esclusa con il parametro `unbuffered` in `get_data` e `get_raw_data`.

Decoding the data is controlled by the static dictionaries `_reading_format`, `lo_fix_format` and `fixed_format`. The keys are names of data items and the values can be an `(offset, type, multiplier)` tuple or another dictionary. So, for example, the _reading_format dictionary entry `'rain' : (13, 'us', 0.3)` means that the rain value is an unsigned short (two bytes), 13 bytes from the start of the block, and should be multiplied by 0.3 to get a useful value.

L'uso di dizionari annidati nel dizionario `fixed_format` permette utili sottoinsiemi di dati per essere decodificati. Ad esempio, per decodificare l'intero blocco `get_fixed_block` viene chiamato senza parametri

```
ws = WeatherStation.weather_station()
print ws.get_fixed_block()
```

Per ottenere la temperatura esterna minima memorizzata, è chiamato `get_fixed_block` con una sequenza di comandi:

```
ws = WeatherStation.weather_station()
print ws.get_fixed_block(['min', 'temp_out', 'val'])
```

Spesso non non c'è nessun obbligo di leggere e decodificare l'intero blocco, siccome i primi 64 byte contengono i dati più utili: l'intervallo dei valori memorizzati, l'indirizzo del buffer dove è memorizzata la lettura corrente e l'ora della data corrente. Il metodo `get_lo_fix_block` fornisce facile accesso a questi dati.

Per ulteriori esempi di utilizzo del modulo WeatherStation, consultare il programma TestWeatherStation.

Dettagli API

Funzioni

decode_status(status)	

Classi

CUSBDrive()	Interfaccia di basso livello della stazione meteo tramite USB.
DriftingClock(logger, name, status, period, ...)	
weather_station([ws_type, status, avoid])	Classe che rappresenta la stazione meteorologica nel programma utente.

pywws.WeatherStation.**decode_status**(*status*)

class pywws.WeatherStation.**CUSBDrive**

Interfaccia di basso livello della stazione meteo tramite USB.

Liberamente ispirato su una classe C++ ottenuta da http://site.ambientweatherstore.com/easyweather/ws_1080_2080_protocol.zip.Non so la provenienza di questo, ma sembra che siano venuti dal produttore.

EndMark = 32

ReadCommand = 161

WriteCommand = 160

WriteCommandWord = 162

read_block(*address*)
Legge 32 byte dalla stazione meteo.

Se la lettura non riesce per qualche motivo, None viene restituito.

Parametri **address** (*int*) – Indirizzo di lettura da.

Ritorna I dati della stazione meteo.

Tipo di ritorno list(int)

write_byte(*address*, *data*)
Scrive un singolo byte sulla stazione meteo.

Parametri

- **address** (*int*) – Indirizzo di scrittura su.
- **data** (*int*) – il valore da scrivere.

Ritorna eseguito con successo.

Tipo di ritorno bool

class pywws.WeatherStation.**DriftingClock**(*logger*, *name*, *status*, *period*, *margin*)

before(*now*)

avoid()

set_clock(*now*)

invalidate()

class pywws.WeatherStation.**weather_station**(*ws_type='1080'*, *status=None*, *avoid=3.0*)
Classe che rappresenta la stazione meteorologica nel programma utente.

Connettersi alla stazione metereologica e preparare per la lettura dei dati.

min_pause = 0.5

margin = 0.9

live_data(*logged_only=False*)

inc_ptr(*ptr*)
Ottenere il puntatore ai dati successivi del buffer circolare.

dec_ptr(*ptr*)
Ottenere il puntatore ai dati precedenti nel buffer circolare.

get_raw_data(*ptr*, *unbuffered=False*)
Ottiene i dati grezzi (raw) dal buffer circolare.

Se "unbuffered" è false, potrebbe essere restituito un valore memorizzato nella cache, che è stato ottenuto in precedenza.

get_data(*ptr*, *unbuffered=False*)
Ottiene i dati decodificati dal buffer circolare.

Se "unbuffered" è false, potrebbe essere restituito un valore memorizzato nella cache, che è stato ottenuto in precedenza.

current_pos()
Ottiene la posizione del buffer circolare dove sono scritti dati correnti.

get_raw_fixed_block (*unbuffered=False*)
Ottiene il grezzo (raw) «fixed block» di impostazioni e dati di min/max.

get_fixed_block (*keys=[]*, *unbuffered=False*)
Ottiene decodificato «fixed block» di impostazioni e dati di min/max.

Per selezionare un sottoinsieme di tutto il blocco tasti.

write_data (*data*)
Scrive una serie di byte singoli sulla stazione meteo. Dati devono essere una matrice di coppie (ptr, value).

```
lo_fix_format = {'alarm_1':  (21, 'bf', ('bit0', 'time', 'wind_dir', 'bit3', 'hum_in_l
fixed_format = {'alarm_1':  (21, 'bf', ('bit0', 'time', 'wind_dir', 'bit3', 'hum_in_lo
data_start = 256
reading_len = {'3080':  20, '1080':  16}
```

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.device_libusb1

Low level USB interface to weather station, using python-libusb1.

Introduzione

This module handles low level communication with the weather station via the python-libusb1 library. It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Collaudo

Run *pywws-testweatherstation* with increased verbosity so it reports which USB device access module is being used:

```
pywws-testweatherstation -vv
11:30:35:pywws.Logger:pywws version 15.01.0
11:30:35:pywws.Logger:Python version 3.3.5 (default, Mar 27 2014, 17:16:46) [GCC]
11:30:35:pywws.WeatherStation.CUSBDrive:using pywws.device_libusb1
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 7f 00␣
↪f0 0f 00 50 04
0020 f9 25 74 26 00 00 00 00 00 00 00 15 01 15 11 31 41 23 c8 00 00 00 46 2d 2c 01 64␣
↪80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00␣
↪00 00 00 00 00
0060 00 00 5a 0a 63 0a 41 01 70 00 dc 01 08 81 dc 01 c5 81 68 01 75 81 95 28 e0 25 24␣
↪29 d9 25 fd 02
0080 b9 02 f4 ff fd ff 85 ff 91 ff 6c 09 00 14 10 19 06 29 12 02 01 19 32 11 09 09 05␣
↪18 12 03 28 13
00a0 00 13 07 19 18 28 13 01 18 23 21 13 09 24 13 02 13 09 24 13 33 13 09 24 13 02 12␣
↪07 28 12 50 13
```

```
00c0 09 24 13 02 13 10 14 16 18 12 02 07 19 00 14 02 14 22 39 13 01 04 10 28 15 01 15␣
↪03 48 12 03 10
00e0 22 02 13 01 30 21 24 12 07 28 11 59 13 03 06 06 43 12 04 13 00 04 12 04 13 00 04␣
↪12 07 31 03 34
```

API

Classi

USBDevice(idVendor, idProduct)	Low level USB device access via python-libusb1 library.

class pywws.device_libusb1.**USBDevice**(*idVendor*, *idProduct*)

Low level USB device access via python-libusb1 library.

Parametri

- **idVendor** (*int*) – the USB «vendor ID» number, per esempio 0x1941.
- **idProduct** (*int*) – the USB «product ID» number, per esempio 0x8021.

read_data(*size*)

Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)

Inviare dati al dispositivo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **buf** (*list(int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.device_pyusb1

Low level USB interface to weather station, using PyUSB v1.0.

Introduzione

This module handles low level communication with the weather station via the PyUSB library (version 1.0). It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Collaudo

Run *pywws.TestWeatherStation* with increased verbosity so it reports which USB device access module is being used:

```
python -m pywws.TestWeatherStation -vv
18:28:09:pywws.WeatherStation.CUSBDrive:using pywws.device_pyusb1
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 00 00␣
↪0f 05 00 e0 51
0020 03 27 ce 27 00 00 00 00 00 00 00 12 02 14 18 27 41 23 c8 00 00 00 46 2d 2c 01 64␣
↪80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00␣
↪00 00 00 00 00
0060 00 00 49 0a 63 12 05 01 7f 00 36 01 60 80 36 01 60 80 bc 00 7b 80 95 28 12 26 6c␣
↪28 25 26 c8 01
0080 1d 02 d8 00 de 00 ff 00 ff 00 ff 00 00 11 10 06 01 29 12 02 01 19 32 11 09 09 05␣
↪18 12 01 22 13
00a0 14 11 11 04 15 04 11 12 17 05 12 11 09 02 15 26 12 02 11 07 05 11 09 02 15 26 12␣
↪02 11 07 05 11
00c0 09 10 09 12 12 02 02 12 38 12 02 07 19 00 11 12 16 03 27 12 02 03 11 00 11 12 16␣
↪03 27 11 12 26
00e0 21 32 11 12 26 21 32 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57␣
↪12 02 06 19 57
```

API

Classi

USBDevice(idVendor, idProduct)	Basso livello del dispositivo di accesso USB tramite libreria PyUSB 1.0.

class pywws.device_pyusb1.**USBDevice**(*idVendor*, *idProduct*)

Basso livello del dispositivo di accesso USB tramite libreria PyUSB 1.0.

Parametri

- **idVendor** (*int*) – the USB «vendor ID» number, per esempio 0x1941.
- **idProduct** (*int*) – the USB «product ID» number, per esempio 0x8021.

read_data(*size*)

Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)

Inviare dati al dispositivo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **buf** (*list(int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.device_pyusb

Low level USB interface to weather station, using PyUSB v0.4.

Introduzione

This module handles low level communication with the weather station via the PyUSB library. It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Collaudo

Run *pywws.TestWeatherStation* with increased verbosity so it reports which USB device access module is being used:

```
python -m pywws.TestWeatherStation -vv
18:28:09:pywws.WeatherStation.CUSBDrive:using pywws.device_pyusb
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 00 00␣
↪0f 05 00 e0 51
0020 03 27 ce 27 00 00 00 00 00 00 00 12 02 14 18 27 41 23 c8 00 00 00 46 2d 2c 01 64␣
↪80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00␣
↪00 00 00 00 00
0060 00 00 49 0a 63 12 05 01 7f 00 36 01 60 80 36 01 60 80 bc 00 7b 80 95 28 12 26 6c␣
↪28 25 26 c8 01
0080 1d 02 d8 00 de 00 ff 00 ff 00 ff 00 00 11 10 06 01 29 12 02 01 19 32 11 09 09 05␣
↪18 12 01 22 13
00a0 14 11 11 04 15 04 11 12 17 05 12 11 09 02 15 26 12 02 11 07 05 11 09 02 15 26 12␣
↪02 11 07 05 11
00c0 09 10 09 12 12 02 02 12 38 12 02 07 19 00 11 12 16 03 27 12 02 03 11 00 11 12 16␣
↪03 27 11 12 26
00e0 21 32 11 12 26 21 32 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57␣
↪12 02 06 19 57
```

API

Classi

USBDevice(idVendor, idProduct)	Basso livello di accesso al dispositivo USB tramite libreria PyUSB.

class pywws.device_pyusb.**USBDevice**(*idVendor*, *idProduct*)

Basso livello di accesso al dispositivo USB tramite libreria PyUSB.

Parametri

- **idVendor** (*int*) – the USB «vendor ID» number, per esempio 0x1941.
- **idProduct** (*int*) – the USB «product ID» number, per esempio 0x8021.

read_data(*size*)
Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)
Inviare dati al dispositivo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **buf** (*list(int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.device_ctypes_hidapi

Interfaccia di basso livello USB della stazione meteo, utilizzando ctypes per accedere a hidapi.

Introduzione

This module handles low level communication with the weather station via ctypes and the hidapi library. It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Collaudo

Run *pywws.TestWeatherStation* with increased verbosity so it reports which USB device access module is being used:

```
python -m pywws.TestWeatherStation -vv
18:10:27:pywws.WeatherStation.CUSBDrive:using pywws.device_ctypes_hidapi
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00␣
↪07 01 00 d0 56
0020 61 1c 61 1c 00 00 00 00 00 00 00 12 02 14 18 09 41 23 c8 00 32 80 47 2d 2c 01 2c␣
↪81 5e 01 1e 80
0040 a0 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 00 00 00␣
↪00 00 00 00 00
0060 00 00 54 1c 63 0a 2f 01 71 00 7a 01 59 80 7a 01 59 80 e4 00 f5 ff 69 54 00 00 fe␣
↪ff 00 00 b3 01
0080 0c 02 d0 ff d3 ff 5a 24 d2 24 dc 17 00 11 09 06 15 40 10 03 07 22 18 10 08 11 08␣
↪30 11 03 07 12
```

```
00a0 36 08 07 24 17 17 11 02 28 10 10 09 06 30 14 29 12 02 11 06 57 09 06 30 14 29 12␣
↪02 11 06 57 08
00c0 08 31 14 30 12 02 14 18 04 12 02 01 10 12 11 09 13 17 19 11 08 21 16 53 11 09 13␣
↪17 19 12 01 18
00e0 07 17 10 02 22 11 06 11 11 06 13 12 11 11 06 13 12 11 11 10 11 38 11 11 10 11 38␣
↪10 02 22 14 43
```

API

Classi

USBDevice(vendor_id, product_id)	Basso livello di accesso al dispositivo USB tramite la libreria hidapi.

class pywws.device_ctypes_hidapi.**USBDevice**(*vendor_id*, *product_id*)

Basso livello di accesso al dispositivo USB tramite la libreria hidapi.

Parametri

- **idVendor** (*int*) – the USB «vendor ID» number, per esempio 0x1941.
- **idProduct** (*int*) – the USB «product ID» number, per esempio 0x8021.

read_data(*size*)

Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, IOError l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)

Inviare dati al dispositivo.

Parametri **buf** (*list(int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.device_cython_hidapi

Interfaccia di basso livello USB della stazione meteo, tramite cython-hidapi.

Introduzione

This module handles low level communication with the weather station via the cython-hidapi library. It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Collaudo

Run *pywws.TestWeatherStation* with increased verbosity so it reports which USB device access module is being used:

```
python -m pywws.TestWeatherStation -vv
18:10:27:pywws.WeatherStation.CUSBDrive:using pywws.device_cython_hidapi
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00␣
↪07 01 00 d0 56
0020 61 1c 61 1c 00 00 00 00 00 00 00 12 02 14 18 09 41 23 c8 00 32 80 47 2d 2c 01 2c␣
↪81 5e 01 1e 80
0040 a0 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 00 00 00␣
↪00 00 00 00 00
0060 00 00 54 1c 63 0a 2f 01 71 00 7a 01 59 80 7a 01 59 80 e4 00 f5 ff 69 54 00 00 fe␣
↪ff 00 00 b3 01
0080 0c 02 d0 ff d3 ff 5a 24 d2 24 dc 17 00 11 09 06 15 40 10 03 07 22 18 10 08 11 08␣
↪30 11 03 07 12
00a0 36 08 07 24 17 17 11 02 28 10 10 09 06 30 14 29 12 02 11 06 57 09 06 30 14 29 12␣
↪02 11 06 57 08
00c0 08 31 14 30 12 02 14 18 04 12 02 01 10 12 11 09 13 17 19 11 08 21 16 53 11 09 13␣
↪17 19 12 01 18
00e0 07 17 10 02 22 11 06 11 11 06 13 12 11 11 06 13 12 11 11 10 11 38 11 11 10 11 38␣
↪10 02 22 14 43
```

API

Classi

USBDevice(idVendor, idProduct)	Basso livello di accesso al dispositivo USB tramite libreria cython-hidapi.

class pywws.device_cython_hidapi.**USBDevice**(*idVendor*, *idProduct*)

Basso livello di accesso al dispositivo USB tramite libreria cython-hidapi.

Parametri

- **idVendor** (*int*) – the USB «vendor ID» number, per esempio 0x1941.
- **idProduct** (*int*) – the USB «product ID» number, per esempio 0x8021.

read_data(*size*)

Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)

Inviare dati al dispositivo.

Parametri **buf** (*list (int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.DataStore

DataStore.py - memorizza le letture in file di facile accesso

Introduzione

Questo modulo è il cuore del mio software per la stazione meteo. Esso memorizza i dati su disco, ma senza il sovraccarico di un complesso database. L'ho progettato per funzionare su una macchina con poca memoria come il mio router Asus. Per ridurre al minimo l'utilizzo della memoria, carica solo l'equivalente di un giorno di dati alla volta in memoria.

Da un punto di vista «user», i dati vengono eseguiti come un incrocio tra un elenco e un dizionario. Ogni record di dati è indicizzato da un oggetto `datetime.datetime` (comportamento di dizionario), ma i record vengono archiviati in ordine ed è possibile accedervi come fogli (comportamento di elenco).

Ad esempio, per accedere ai dati orari per il giorno di Natale 2009, uno potrebbe effettuare le seguenti operazioni:

```
from datetime import datetime
from pywws import DataStore
hourly = DataStore.hourly_store('weather_data')
for data in hourly[datetime(2009, 12, 25):datetime(2009, 12, 26)]:
    print data['idx'], data['temp_out']
```

Alcuni esempi di accesso dati:

```
# get value nearest 9:30 on Christmas day 2008
data[data.nearest(datetime(2008, 12, 25, 9, 30))]
# get entire array, equivalent to data[:]
data[datetime.min:datetime.max]
# get last 12 hours worth of data
data[datetime.utcnow() - timedelta(hours=12):]
```

Si noti che l'indice `datetime.datetime` è in formato UTC. Potrebbe essere necessario applicare un offset per convertire in ora locale.

Il modulo fornisce cinque classi per archiviare dati diversi. *`data_store`* prende i dati «raw» dalla stazione meteo; *`calib_store`*, *`hourly_store`*, *`daily_store`* e *`monthly_store`* memorizza i dati elaborati (vedi `pywws.Process`).Tutti e tre sono derivati dallo stessa classe `core_store`, differiscono solo per le chiavi e i tipi di dati memorizzati in ogni record.

Dettagli API

Funzioni

`safestrptime`(date_string[, format])

Classi

ParamStore(root_dir, file_name)	
calib_store(root_dir)	Memorizza i dati "calibrati" della Stazione Meteo.
core_store(root_dir)	
daily_store(root_dir)	Memorizza i dati giornalieri di riepilogo della Stazione Meteo.
data_store(root_dir)	Memorizza i dati grezzi Stazione Meteo.
hourly_store(root_dir)	Memorizza i dati orari di riepilogo della Stazione Meteo.
monthly_store(root_dir)	Memorizza i dati mensili di riepilogo della Stazione Meteo.
params(root_dir)	I parametri sono memorizzati in un file «weather.ini» in root_dir.
status(root_dir)	Lo stato viene memorizzato in un file «status.ini» in root_dir.

pywws.DataStore.**safestrptime**(*date_string*, *format=None*)

class pywws.DataStore.**ParamStore**(*root_dir*, *file_name*)

flush()

get(*section*, *option*, *default=None*)
Ottiene un dato di parametro e restituisce una stringa.

Se si specifica default e la sezione o l'opzione non sono definite nel file, sono creati e impostati su default, che è poi il valore restituito.

get_datetime(*section*, *option*, *default=None*)

set(*section*, *option*, *value*)
Impostare l'opzione nella sezione valore stringa.

unset(*section*, *option*)
Rimuovere l'opzione dalla sezione.

class pywws.DataStore.**params**(*root_dir*)
I parametri sono memorizzati in un file «weather.ini» in root_dir.

class pywws.DataStore.**status**(*root_dir*)
Lo stato viene memorizzato in un file «status.ini» in root_dir.

class pywws.DataStore.**core_store**(*root_dir*)

before(*idx*)
Ritorno data ora dell'ultimo record di dati esistenti cui data ora è < idx.

Potrebbe anche non essere nello stesso anno! Se non esiste nessun record, restituisce None.

after(*idx*)
Restuisce data ora del record di dati esistente più antico cui data ora è >= idx.

Potrebbe anche non essere nello stesso anno! Se non esiste nessun record, restituisce None.

nearest(*idx*)
Restuisce data ora del record cui data ora è più vicina di idx.

flush()

class pywws.DataStore.**data_store**(*root_dir*)
Memorizza i dati grezzi Stazione Meteo.

```
key_list = ['idx', 'delay', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure'
conv = {'status':  <type 'int'>, 'wind_ave':  <type 'float'>, 'rain':  <type 'float'>,
```

class pywws.DataStore.**calib_store**(*root_dir*)
Memorizza i dati "calibrati" della Stazione Meteo.

```
key_list = ['idx', 'delay', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure'
conv = {'status':  <type 'int'>, 'wind_ave':  <type 'float'>, 'rain':  <type 'float'>,
```

class pywws.DataStore.**hourly_store**(*root_dir*)
Memorizza i dati orari di riepilogo della Stazione Meteo.

```
key_list = ['idx', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'rel_pr
conv = {'pressure_trend':  <type 'float'>, 'wind_ave':  <type 'float'>, 'rain':  <type
```

class pywws.DataStore.**daily_store**(*root_dir*)
Memorizza i dati giornalieri di riepilogo della Stazione Meteo.

```
key_list = ['idx', 'start', 'hum_out_ave', 'hum_out_min', 'hum_out_min_t', 'hum_out_ma
conv = {'temp_in_min':  <type 'float'>, 'temp_in_max':  <type 'float'>, 'uv_ave':  <ty
```

class pywws.DataStore.**monthly_store**(*root_dir*)
Memorizza i dati mensili di riepilogo della Stazione Meteo.

```
key_list = ['idx', 'start', 'hum_out_ave', 'hum_out_min', 'hum_out_min_t', 'hum_out_ma
conv = {'uv_ave':  <type 'float'>, 'illuminance_max_hi_t':  <function safestrptime>, '
```

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.Localisation

Localisation.py - effetua le traduzioni di stringhe in lingua locale

```
usage: python -m pywws.Localisation [options]
options are:
 -h       or  --help       display this help
 -t code  or  --test code  test use of a language code
```

Introduzione

Alcuni dei moduli pywws, come WindRose.py, può utilizzare automaticamente la lingua locale per cose come la direzione del vento. Il modulo Localisation.py, per lo più copiato da esempi nella documentazione di Python, permette questo

Localizzazione di pywws è fatta in due parti - tradurre le stringhe come "rising very rapidly", e cambiando le impostazioni internazionali che controlla le diciture come i nomi dei mesi e la rappresentazione dei numeri (e.g. "23,2" cambiando in "23.2"). Su alcuni computer potrebbe non essere possibile impostare le impostazioni internazionali, ma è possibile utilizzare le stringhe tradotte.

Usare un linguaggio diverso

Il linguaggio utilizzato da pywws e definito nel file `weather.ini` alla sezione `[config]`. Questo può essere un codice di due lettere della lingua, ad esempio `it` (Italiano), o si può specificare una variante nazionale, come `fr_CA` (Canadian French). Potrebbe anche includere un set di caratteri, ad esempio `de_DE.UTF-8`.

La scelta della lingua è a carico del sistema operativo, così Localisation.py può essere eseguito come programma autonomo per testare i codici di lingua. Un buon punto di partenza potrebbe essere la variabile di ambiente di sistema " LANG ", ad esempio:

```
jim@brains:~/Documents/weather/pywws/code$ echo $LANG
en_GB.UTF-8
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t en_GB.UTF-8
Locale changed from (None, None) to ('en_GB', 'UTF8')
Translation set OK
Locale
  decimal point: 23.2
  date & time: Friday, 14 October (14/10/11 13:02:00)
Translations
  'NNW' => 'NNW'
  'rising very rapidly' => 'rising very rapidly'
  'Rain at times, very unsettled' => 'Rain at times, very unsettled'
jim@brains:~/Documents/weather/pywws/code$
```

Nella maggior parte dei casi è richiesto un codice di non più di due lettere:

```
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t fr
Locale changed from (None, None) to ('fr_FR', 'UTF8')
Translation set OK
Locale
  decimal point: 23,2
  date & time: vendredi, 14 octobre (14/10/2011 13:04:44)
Translations
  'NNW' => 'NNO'
  'rising very rapidly' => 'en hausse très rapide'
  'Rain at times, very unsettled' => 'Quelques précipitations, très perturbé'
jim@brains:~/Documents/weather/pywws/code$
```

Se impostate una lingua non supportata, pywws usa per default l'Inglese:

```
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t ja
Failed to set locale: ja
No translation file found for: ja
Locale
  decimal point: 23.2
  date & time: Friday, 14 October (10/14/11 13:08:49)
Translations
  'NNW' => 'NNW'
  'rising very rapidly' => 'rising very rapidly'
  'Rain at times, very unsettled' => 'Rain at times, very unsettled'
jim@brains:~/Documents/weather/pywws/code$
```

Dopo aver trovato un codice di lingua adatto che funziona, è possibile configurare pywws per usarlo modificando il file `weather.ini`:

```
[config]
language = fr
```

Creazione di una nuova traduzione

Se non c'è nessun file di traduzione per la lingua preferita, allora avete bisogno di crearne uno. Vedi *Come utilizzare pywws in un altra lingua* per istruzioni dettagliate.

Funzioni

SetApplicationLanguage(params)	Impostare le impostazioni internazionali e la traduzione di un programma pywws.
SetLocale(lang)	Impostare le impostazioni locali utilizzate dal programma.
SetTranslation(lang)	Impostare la traduzione utilizzata da (alcuni) moduli pywws.
main([argv])	

pywws.Localisation.**SetLocale**(*lang*)

Impostare le impostazioni locali utilizzate dal programma.

Questo riguarda l'intera applicazione, cambiando il modo in cui le date, valute e numeri sono rappresentati. Essa non deve essere chiamata da una routine di libreria che può essere utilizzata in un altro programma.

Il parametro `lang` può essere qualsiasi stringa che è riconosciuto da `locale.setlocale()`, per esempio `it`, `it_IT` or `it_IT.UTF-8`.

Parametri **lang** (*string*) – codice della lingua.

Ritorna eseguito con successo.

Tipo di ritorno bool

pywws.Localisation.**SetTranslation**(*lang*)

Impostare la traduzione utilizzata da (alcuni) moduli pywws.

Questo imposta l'oggetto di traduzione `Localisation.translation` per utilizzare una particolare lingua.

Il parametro `lang` può essere qualsiasi stringa nel formato `it`, `it_IT` o `it_IT.UTF-8`. Tutto ciò dopo un carattere `.` viene ignorato. Nel caso di una stringa come `it_IT`, la routine cercherà un file di lingua `it_IT` prima di cercare un file `it`.

Parametri **lang** (*string*) – codice della lingua.

Ritorna eseguito con successo.

Tipo di ritorno bool

pywws.Localisation.**SetApplicationLanguage**(*params*)

Impostare le impostazioni internazionali e la traduzione di un programma pywws.

Questa funzione legge la lingua dal file di configurazione, quindi chiama *SetLocale()* e *SetTranslation()*.

Parametri **params** (*object*) – a *pywws.DataStore.params* object.

pywws.Localisation.**main**(*argv=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.Logger

Codice comune per la registrazione di informazioni ed errori.

Funzioni

ApplicationLogger(verbose[, logfile])

pywws.Logger.**ApplicationLogger**(*verbose*, *logfile=None*)

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

pywws.constants

Bits of data used in several places.

This module collects together some "constants" that are used in other pywws modules.

Classi

Twitter

class pywws.constants.**Twitter**

```
consumer_key = '62moSmU9ERTs0LK0g2xHAg'
consumer_secret = 'ygdXpjr0rDagU3dqULPqXF8GFgUOD6zYDapoHAH9ck'
```

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

3.2 Indici e tabelle

- genindex
- modindex
- search

CAPITOLO 4

Ringraziamenti

Non sarei stato in grado di ottenere tutte le informazioni dalla stazione meteo senza avere accesso ai sorgenti di Michael Pendec's programma «wwsr». Sono anche grata alla Dave Wells per la decodifica del weather station's «fixed block» data.

Infine, un grande ringraziamento a tutti gli utenti pywws che hanno aiutato con domande e suggerimenti e soprattutto a coloro che hanno tradotto pywws e la relativa documentazione in altre lingue.

CAPITOLO 5

Licenze

pywws - Python software per Stazione Meteo USB senza filo.
http://github.com/jim-easterbrook/pywws

Comments or questions? Please subscribe to the pywws mailing list http://groups.google.com/group/pywws and let us know.

Indice del modulo Python

p

Indice

W

Z